НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:

Завідувач кафедри

_____ Олександр Коваль

«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні технології моніторингу довкілля»

спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Веб система генерації гідроакустичного сигналу за променевою моделлю»

Виконав:
студент IV курсу, групи ТМ-62
Повєткін Дмитро Олексійович _____

Керівник:
Ст. в. Гайдаржи Володимир Іванович _____

Рецензент: _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
"Київський політехнічний інститут імені Ігоря Сікорського"

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 Комп'ютерні науки та інформаційні технології

Спеціалізація Інформаційні технології моніторингу довкілля

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
"___"_____2020р.

ЗАВДАННЯ

на дипломну роботу студенту
Повєткін Дмитро Олексійович
(прізвище, ім'я, по батькові)

1. Тема роботи «Веб система генерації гідроакустичного сигналу за променевою моделлю»

керівник роботи Гайдаржи Володимир Іванович
(прізвище, ім'я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від "__"__ ____р. № _____

2. Строк подання студентом роботи 3.06.2020

3. Вихідні дані до роботи платформа VS Code, мова програмування Javascript з быблыотекою React для клієнтської сторони та мова програмування Node.js для серверної

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Розробити алгоритм для моделювання гідроакустичного сигналу. Реалізувати код веб системи для цього алгоритму. Створити зручний користувацький інтерфейс для вводу початкових данних та отримання результату у зручному форматі. Візуально зобразити сигнал у вигляді графіка та згенерувати файл json розширення що буде містити дані цього сигналу для подальшого прослуховування його користувачем.

5. Перелік ілюстративного матеріалу

«Постановка задачі», «Сфери застосування даного рішення», «Існуючі рішення», «Використані технології», «Архітектура системи», «Головний екран веб системи», «Головні елементи інтерфейсу», «Введення нових направлень», «Відображення об’єктів на мапі», «Кінцевий результат», «Висновки»

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”14” жовтня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	14.10.2019	
2.	Вивчення та аналіз задачі	14.10.2019-23.12.2019	
3.	Розробка архітектури та загальної структури системи	02.02.2020-03.03.2020	
4.	Розробка структур окремих підсистем	04.03.2020-14.04.2020	
5.	Програмна реалізація системи	15.04.2020-19.05.2020	
6.	Оформлення пояснювальної записки	20.05.2020-03.06.2020	
7.	Захист програмного продукту	30.05.2020	
8.	Передзахист	7.06.2020	
9.	Захист	18.06.2020	

Студент _____ Дмитро ПОВЄТКІН
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Володими ГАЙДАРЖИ
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Дана дипломна робота присвячена розробці веб системи для моделювання гідроакустичного сигналу променевою моделлю.

Для досягнення мети були вирішені наступні задачі:

1. Проведено аналіз необхідної функціональності системи.

2. Спроектовано чітку та узгоджену архітектуру веб системи.

3. Розроблено API з дотриманням REST принципів.

4. Розроблено та протестовано систему.

5. Веб систему розгорнуто в браузері.

Обсяг звіту становить 44 сторінок, міститься 23 ілюстрацій. Загалом опрацьовано 14 джерел.

ABSTRACT

This thesis is devoted to the development of a web system for modeling the sonar signal by a beam model.

To achieve this goal, the following tasks were solved:

1. The analysis of the necessary functionality of the system is carried out.
2. A clear and coherent web system architecture is designed.
3. Developed API with REST principles.
4. The system is developed and tested.
5. The web system is deployed in a browser.

Scope of the report is 44 pages contain 23 illustrations. Generally 14 sources have been processed.

ОГЛАВЛЕНИЕ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

UI	–	User interface
REST	–	Representational State Transfer,
API	–	Application Program Interface
JSON	–	JavaScript Object Notation
XML	–	eXtensible Markup Language

ВСТУП

В наш час гідроакустика знайшла широке використання у багатьох сферах пов’язаних із водним середовищем, а саме гідролокація у будь-яких масштабах, від ехолотів для рибалки до локації морських об’єктів у військових діях, зв’язок під водоймами, розпізнавання рельєфу дна океанів та морів. В даний час Світовий океан вносить все більш помітний внесок в науково-практичну діяльність людини. Вчені, вивчаючи характер дна і донних відкладень - цього найціннішого літописного матеріалу з історії Землі, отримують відповіді на принципові питання геології, географії, геофізики, геохімії, біології та інших наук. З океану людство добуває їжу, прісну воду, сировину, корисні копалини, освоює його енергетичні ресурси. Популярність використання гідроакустичних сигналів, які були зазначені вище полягає в нестандартній поведінці звуку під водоймами. В той час як більшість відомих людству методів передачі інформації не є ефективними в водному середовищі, звукові хвилі, через властивість низького затухання, зарекомендували себе як дуже ефективний метод транспортування підводної інформації. Звук у воді поширюється нелінійно, зазнаючи вигини (рефракція), відображення, додавання і віднімання відбитих копій сигналу (реверберація і багатопроменеве поширення), доплеровське зміщення а також розтягування і стиснення спектра. Відомо, що в водному середовищі, звуки поширюються на більші відстані ніж у повітрі. Для вимірювань звуку у водному середовищі використовується пристрій - гідрофон, який підводним еквівалентом мікрофона. Даний пристрій призначений для запису або прослуховування звуку під водою.Також гідроакустика використовується в гідроакустичному зв'язоку і телеметрії. Засоби гідроакустичної зв'язку - це пристрої, призначені для обміну інформацією передачею сигналів у воді по гідроакустичного каналу. За допомогою цих пристроїв з'явилася можливість здійснювати телефонний і телеграфний зв'язок і передачу зображень Для проведення натурного експерименту використовують багато ресурсів та вимагають додаткового виконання процедур: фіксації умов експерименту, тимчасової прив'язки подій і ретельного їх опису для зберігання і подальшого поглибленого аналізу і обробки. Цей сервіс є частиною системи, що може

отримувати й ідентифікувати сигнали різних водних об’єктів з різних водних середовищ. Для розробки програмного забезпечення було використано мову програмування Node.js[14] для написання серверної частини і розрахунків, середовище програмування VS Code, а також мову програмування JavaScript[9] з бібліотекою React для створення користувацького інтерфейсу.

1. ЗАДАЧА РОЗРОБКИ ВЕБ СИСТЕМИ ДЛЯ МОДЕЛЮВАННЯ ГІДРОАКУСТИЧНОГО СИГНАЛУ

Веб-система для генерації гідроакустичного сигналу надає користувачам змогу проводити моделювання схеми. Також користувачі мають змогу вводити данні для моделювання, зберігання, редагування та видалення змодельованних даних в системі, а також відображення результатів на графіках

1.1 Задачі поставлені перед системою

До задач розроблюваної веб системи були такі наступні вимоги:

1. Визначення основних параметрів морського об’єкта;
2. Генерація профілю морського дна;
3. Побудова траєкторії руху морських об’єктів;
4. Генерація гідроакустичного сигналу;
5. Візуалізацію характеристик отриманого сигналу (P, Vx, Vy, Vz);
6. Формування json файлу з шуканим сигналом.

1.2 Компоненти програмної системи

Для виконання поставлених перед системою задач передбачається розробка наступних компонентів та підсистем:

— Веб-інтерфейс користувача у браузері.

— Веб-сервіс для розрахунку моделі.

— База даних для збереження результатів .

— Сервер для веб-систем.

У веб інтерфейсі розташовані поля для вводу даних та графіки для відображення результатів моделювання, також взаємодіє з веб-сервісом для моделювання результату. Також у користувацькому інтерфейсі присутня можливість

для завантаження файлу з результатами розрахунків для подальшого використання у інших системах. Система має бути легкою у використані та зрозумілою для будь-якого користувача. Має працювати швидко, щоб користувач міг швидко проводити моделювання

Сервіс для розрахунку моделі призначений для паралельного розрахунку моделі для відображення даних на веб-інтерфейсі, також веб-сервіс взаємодіє з базою даних для збереження результату для подальшого використання, а ще сервіс створює файл з розширенням json з усіма розрахунками для подальшого його використання у інших системах. Для того щоб система працювала швидко ми використовуємо асинхронні методи для проведення розрахунків

База даних виконує роль сховища для зберігання даних на сервері, призначений для зберігання даних для подальшого використання, це потрібно для того щоб мати можливість працювати з даними з попереднього сеансу

Сервер використовується для об’єднання всієї системи та їх координування, на ньому також розташована база даних, система розрахунків, веб-інтерфейс. Він є основою для всієї системи

1.3 Потенційні користувачі

Потенційні користувачі веб-системи, розробленої під час виконання даної роботи – це виробники локаторів, військові, а також науковці які займаються вивченням рельєфу океанів.

Ця система дуже корисна для науковців, через такі причини:

— Використання системи для моделювання будь якої ситуацій.
— Система економить ресурси, так як не потребую коштів для випробувань.
— Дає можливість взяти данні для використання у своїх дослідах

Система орієнтована будь який сегмент ринку, так як і для професіоналів так і для любителів. Для любителів це можливість проводити експеременти не купляючи спеціальне обладнання, а для професіналів моделювання швидких дослідів без потреби використання преборів

2. ОПИС КОНЦЕПЦІЇ ГІДРОАКУСТИЧНОГО СИГНАЛУ

2.1 Опис концепції гідроакустичного сигналу

Гідроакустика - вивчення і застосування звуку у воді. Гідроакустика, використовуючи гідроакустичну технологію, найчастіше використовується для моніторингу підводних фізичних і біологічних характеристик [2].

Термін гідроакустика описує вивчення звукових хвиль у воді та її застосування. Гідроакустичний моніторинг передбачає запис сигналів, які показують зміни тиску води, що створюються звуковими хвилями у водоймі. Звук поширюється дуже швидко через воду, так що його можна почути і виявити на дуже великих відстанях. У воді є один шар, де звуковий рух є більш повільним, але дуже ефективним. Цей шар є звукоізоляційним і ранговим каналом SOFAR, який зазвичай знаходиться на глибині 1000 м.

Гідроакустичні технології спочатку розвивалися на початку 20-го століття з метою підвищення безпеки морських подорожей. Звукові хвилі випромінювалися, а їх відбиття вимірювалися об'єктами, такими як айсберги та мілини у воді.

Гідроакустика може бути використана для виявлення глибини водного тіла (батиметрія). Існує багато причин шуму від судноплавства. Вони можуть бути поділені на ті, які викликані пропелером, ті, що викликаються машиною. Для зондування дна вод (батиметрія) лазерним випромінюванням використовується лазер видимого діапазону, зеленого або синього кольору. Мінімум поглинання води знаходиться в синій частині спектра, проте в реальних умовах через розчинених у воді речовин мінімум зміщений в зелену область спектра.

Гідроакустика забезпечує радіус огляду приблизно в два рази більший, ніж глибина. Це означає, що охоплення рельєфу дна обмежений на мілководді і в

прибережних зонах. З водного транспорту гідроакустичному методами важко сканувати перемежовуються мілини і мілководді.

2.2 Технології гідроакустики

Звукові хвилі — єдиний вид хвиль, які мають можливість розповсюджуватися в морському середовищі без великого ослаблення на значні відстані, тому на їх основі створюються дистанційні методі і технології вивчення рельєфу дна, дослідження складу і різноманітних властивостей морських ґрунтів і порід, визначення глибин і виявлення рухомих і нерухомих об’єктів, пошук місцезнаходження затонулих об’єктів за заданими характеристик, перевірка стану підводних гідротехнічних споруд, дослідження стану підводних частин гідротехнічних споруд, дослідження середовища та ін. [3].

В більшості гідроакустичні системи мають дуже схожі складові та використовуються у вирішенні багатьох задач, що призводить до зацікавленості в взаємодії між різними організаціями, які так або інакше, працюють в напрямах застосування та дослідження гідроакустичних технологій. [4].

Гідроакустичні засоби (датчиків) створюються для виконання певних цілей, наприклад:

— геоакустичні;

— знаходження ресурсів(сировинних і біологічних);

— моніторингу стратифікації морського середовища;

— розпізнання підводних ворожих сил;

— робота гідроакустичних станцій та систем підводних та надводних кораблів;

— робота гідроакустичних систем розвідки ситуації в різних підводних зонах.

2.3 Технології генерування сигналу

Пристрій для генерування гідроакустичних сигналів, що містить генератор сигналу, в який входять послідовно з'єднані помножувальний цифро-аналоговий перетворювач та фільтр нижніх частот, який відрізняється тим, що в пристрої для генерування гідроакустичних сигналів введений фільтр Найквіста, а у вищевказаний генератор сигналу додатково введені синтезатор несучого коливання ультразвукової частоти та схема формування амплітуди ультразвукового сигналу, що змінюється по закону sin ft/ft, де f - частота ультразвукового сигналу, t - час, причому до одного входу цифро-аналогового перетворювача підключено синтезатор несучого коливання ультразвукової частоти, а до другого входу - схему формування амплітуди ультразвукового сигналу, вихід фільтра низьких частот з'єднаний з входом фільтра Найквіста, вихід якого є виходом пристрою.

2.4 Розповсюдження сигналу у воді

Вимірювання акустичних сигналів можливі, якщо їх амплітуда перевищує мінімальний поріг, який визначається частково за рахунок обробки сигналу, що використовується і частково за рівнем фонового шуму. Навколишній шум є те, що частина отриманого шуму, який не залежить від джерела, приймача і платформ характеристик.

Фоновий шум присутній в океані, або навколишнього шуму, має багато різних джерел, і змінюється в залежності від місця розташування і частоти. При найнижчих частотах, приблизно від 0,1 Гц до 10 Гц, океанської турбулентності і мікросейсмі є основним внесок в тлі шуму. Типові рівні шуму спектра зменшується зі збільшенням частоти приблизно від 140 дБ на 1 μPa² / Гц при 1 Гц до приблизно 30 дБ на 1 μPa² / Гц при 100 кГц. Віддалений рух суден є одним з домінуючих джерел шуму в більшості областей для частот районі 100 Гц, в той час як вітровий шум поверхні є основним джерелом між 1 кГц і 30 кГц.

2.5 Існуючі системи для генерації гідроакустичного сигналу

Lucy Software — це програмне забезпечення, розроблене для персональних комп'ютерів та ноутбуків, яке дозволяє переглядати та взаємодіяти з даними, які були зібрані з гідрофонів icListen Smart Hydrophones. Будування графіків за допомогою Lucy дозволяє проводити точні вимірювання з записаних або отриманих у реальному часі даних.[8]

Переваги програмного продукту Lucy Software

1. Взаємодіють з усіма продуктами IcList Smart Hydrophone .
2. Існує можливість контролювати параметри збору даних у гідрофоні .
3. Існує можливість отримувати акустичні дані в реальному часі з icListen та переглядати їх.
4. Одночасна взаємодія з багатьма сигналами .
5. Отримання чіткого сигналу фільтруванням даних з допомогою спеціального інструменту для скасування шуму Lucy.

Області застосування програми Lucy:

1. Вимірювання шумів у водних середовищах .
2. Моніторинг об'єктів під водою .
3. Наукові та океанографічні дослідження .
4. Моніторинг життєдіяльності морських тварин .
5. Екологічний моніторинг.

Режими роботи Lucy Software:

1. Режим реального часу .
   - **A.** Візуальне відображення акустичних даних, під час вимірювання.
   - **B.** Демонстрація часових рядів FFT.
   - **C.** Збереження початкових даних для подольшого використання.
2. Режим скасування шуму.

A. Можливість переглядати шуми під час моделювання та подальше видалення його з дисплею.

B. Початкові дані є константами.

3. Режим відтворення шуму .

A. Візуальне відображення зібраних акустичних даних.

B. Перегляд FFT та даних сигналу;

3. ЗАСОБИ РОЗРОБКИ

При розробці програмного продукту були використані такі технології:

Середовищем розробки було обрано Visual Studio Code через зручний інтерфейс та велику кількість розширень для поліпшення написання коду.

Для виконання розрахунків була обрана мова програмування Node.js.

Для графічного інтерфейсу було використано мову програмування Javascript з бібліотекою React.

В якості бази даних використовується MongoDB

Взаємодії між клієнтом и сервером використовується протокол HTTP

3.1 Середовище розробки VS Code

Visual Studio Code це середовище для програмування на будь якій мові програмування. Воно дуже зручне для використання через користувацький інтерфейс.

Visual Studio Code має дуже велику кількість розширень для поліпшення роботи. Велику популярність мають розширення зв’язані з базами даних, бо дають можливість робити запити до бази навіть не розуміючи мову запитів SQL чи NoSQL

1. Visual Studio Code підтримує роботу із JavaScript, CSS і HTML
2. Автодоповнення коду для мов програмування JavaScript, HTML і CSS (для ключових слів, тегів, змінних, параметрів та функцій).
3. Live Edit: зміни в коді можна відразу переглянути в браузері навіть без перезавантаження веб-сторінки .
4. Підтримка розмітки стилів CSS/LESS /SCSS/SASS(підсвічування помилок, валідація, автодоповнення коду тощо) .
5. Пошук використань та ініціалізації змінних, функцій тощо .
6. Рефакторинг JavaScript (виділення, перейменування, змінної / функції, переміщення / копіювання, вбудовування функції / змінної).
7. Інтеграція JavaScript з фреймворками, такими як Angular, React, Vue та інші.

8. Робота з системами контролю версій — зручний інтерфейс для додавання змін до віддаленого репозиторію та стягування їх звідти, а також вирішення конфліктів в коді різних версій .

3.2 Node.js

Node.js — платформа з відкритим кодом для виконання високопродуктивних мережних застосунків, написаних мовою JavaScript. Засновником платформи є Раян Дал. Якщо раніше Javascript застосовувався для обробки даних в браузері користувача, то node.js надав можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їх виконання. Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників.

У традиційних мовах програмування (C, Java, Python, PHP) всі інструкції, за замовчуванням, є блокуючими, якщо тільки розробник явно не подбає про асинхронному виконанні коду. В результаті якщо, наприклад, в такому середовищі, зробити мережевий запит для завантаження якогось JSON-коду, виконання потоку, з якого зроблено запит, буде призупинено до тих пір, поки не завершиться отримання і обробка відповіді.

Node.js має наступні властивості:

1. Однопотокова асинхронна модель виконання запитів.
2. Ввід та вивід без застосування блокування.
3. Система з модулів CommonJS.
4. Компілятор JavaScript Google V8.

Для керування модулями використовується менеджер пакетів npm (node package manager).

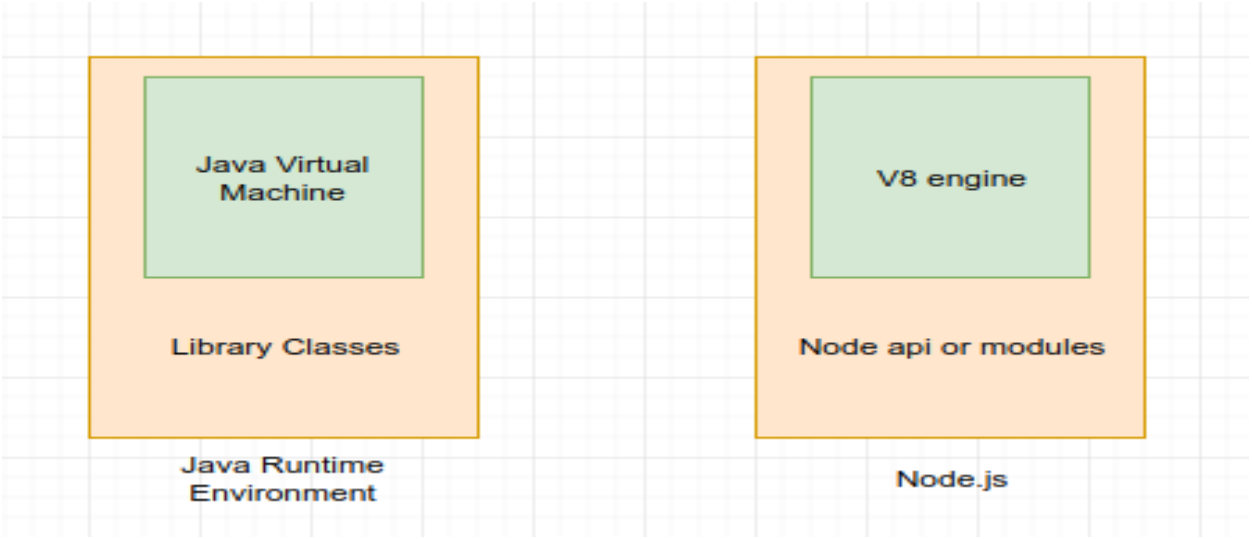


Рисунок 3.1 — Порівняння Java та Node.js

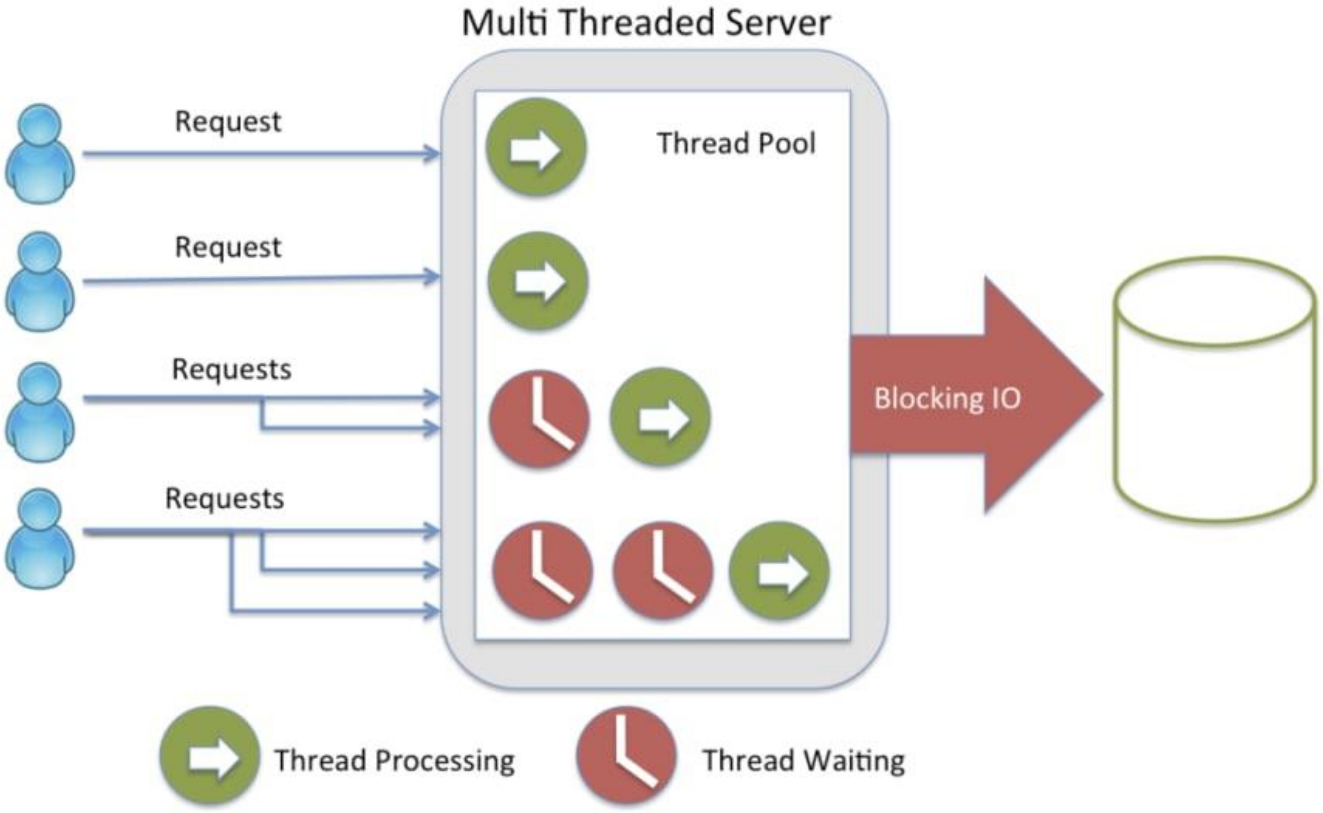


Рисунок 3.2 — Діаграма роботи Node.js

Популярні фреймворки Node.js:

1. Express.js[7] є найпопулярнішим Node.js-фреймворком. Він швидкий, компактний, не нав'язує розробнику жорстких архітектурних рішень. В основі його стрімкого розвитку лежить простота і зрозумілість. Можливо, він ідеологічно ближче всіх інших інструментів до ідей, які лежать в основі, Node, слідуючи яким він є легковажну модульну систему.
2. Koa.js - фреймворк, створений тією ж командою, яка займається Express. Його просувають як “фреймворк наступного покоління для Node.js”. Його можна охарактеризувати як систему, яка відрізняється компактністю, виразністю і надійністю. Koa.js підходить для розробки веб-додатків і API.
3. Meteor, з іншого боку, використовує “чистий” JavaScript і Node.js всередині досить-таки масштабної конструкції. Meteor і сам по собі - це ціла екосистема, яка може підійти для розробки більш складних серверних додатків. Однак, використання Meteor може ускладнитися, якщо потрібно щось, що не вбудовано в систему.

3.3 JavaScript

JavaScript (JS) — об'єктно-орієнтована, прототипна, динамічна мова програмування. Програмна реалізація стандарту ECMAScript. Найчастіше вона використовується для створення веб-сторінок, що надає можливість на стороні клієнта — пристрої користувача, мати контроль над браузером, взаємодіяти з користувачем, асинхронно передавати дані на сервер та отримувати їх від нього, змінювати зовнішній вигляд та структуру веб-сторінки. JavaScript характеризують як скриптову та прототипну мову програмування з динамічними типами. Також JavaScript частково підтримує й інші парадигми програмування, такі як: функціональну та імперативну і деякі властивості архітектури як: слабка та динамічна типізація, керування пам'яттю автоматично, функції як об'єкти першого класу та прототипне наслідування.

Мова JavaScript використовується для:

1. Написання скриптів сторінок для надання їм інтерактивності та динамічності;
2. Створення односторінкових веб-застосунків — SPA (Single Page Application) за допомогою різних фреймворків (React, Angular, Vue);
3. Написання коду на стороні сервера за допомогою платформи Node.js;
4. Створення додатків робочого столу, за допомогою технологій Electron або NW.js;
5. Створення мобільних додатків .

3.4 React

React— відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків. React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC)[13] Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови веб-застосунків. Як бібліотеку інтерфейсу користувача React найчастіше використовують разом з іншими бібліотеками, такими як Redux.

Компонент React - це ділянка коду, який представляє частину веб-сторінки. Кожен компонент - це JavaScript-функція, яка повертає шматок коду, що представляє фрагмент сторінки. Для формування сторінки ми викликаємо ці функції в певному порядку, збираємо разом результати викликів і показуємо їх користувачеві.

Кожен компонент має кілька “методів життєвого циклу”. Перевизначення такого методу дозволяє виконувати код на конкретному етапі цього процесу. Давайте

розглянемо всі три етапи і розберемося в якій послідовності, і які методи викликаються на різних етапах, і як ви можете використовувати ці методи.

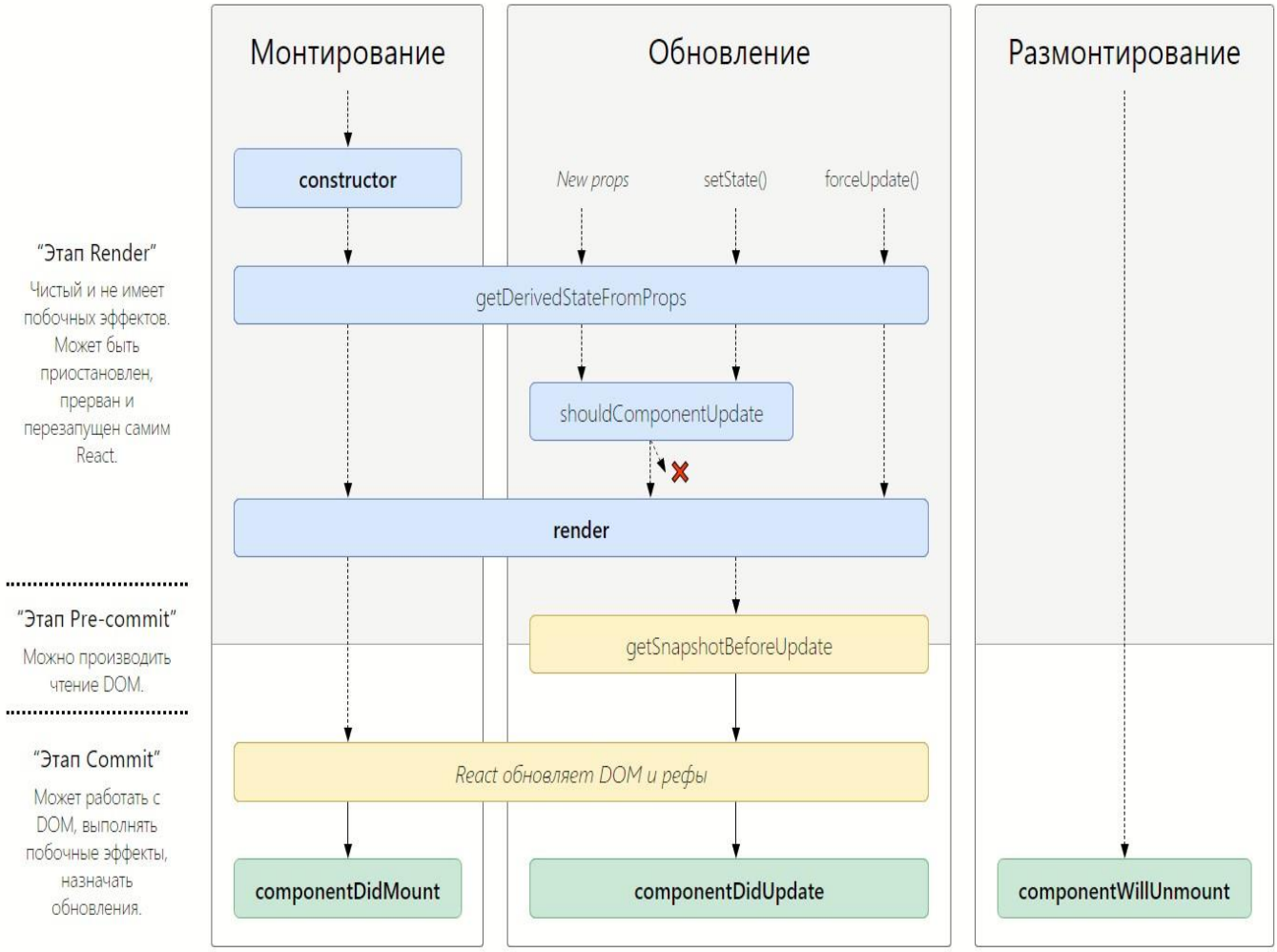


Рисунок 3.3 — Діаграма відображення метод системи

Плюси React:

1. Дуже добре підходить для командної розробки, суворе дотримання UI, і шаблону робочого процесу.
2. Розробка UI на основі окремих компонентів - це майбутнє web-розробки і ви повинні почати робити це вже зараз.

Мінуси React:

1. Якщо ваш додаток чи веб-сайт не насичені великою кількістю динамічних сторінок, вам доведеться писати дуже багато коду, вирішуючи маленькі завдання;
2. React не підтримує браузери від IE8 і молодше, і ніколи не буде;

3.5 MongoDB

MongoDB - це база даних, яка зберігає ваші дані у вигляді документів. Як правило, ці документи мають JSON (* JavaScript Object Notation - текстовий формат обміну даними, заснований на JavaScript.

MongoDB і бази даних документів в цілому вирішують ряд проблем з такими традиційними реляційними базами даних:

1. Сувора схема: з реляційною базою даних, якщо у вас динамічно сформовані дані, ви змушені або створити купу випадкових різних стовпців даних, занести туди Blob-дані або використовувати конфігурацію EAV, у всього цього значні недоліки.
2. Труднощі масштабування: якщо даних настільки багато, що вони не поміщаються на один сервер, MongoDB пропонувала механізми, що дозволяють масштабувати їх на кількох машинах.
3. Складні модифікації схеми: ніяких міграцій! У реляційній базі даних зміна структури БД може стати величезною проблемою (особливо коли даних стає дуже багато). MongoDB змогла значно спростити цей процес. І зробило його настільки легким, що ви можете просто оновлювати схему на ходу і дуже швидко рухатися далі.
4. Продуктивність запису: продуктивність MongoDB була хорошою, особливо при грамотному налаштуванні. Навіть конфігурація MongoDB з коробки, за яку її часто критикували, демонструвала деякі вражаючі показники продуктивності.

Проте існують також такі проблеми:

1. Втрата транзакцій: транзакції є основною особливістю багатьох реляційних баз даних. Транзакційність означає, що ви можете виконувати декілька операцій атомарно і можете гарантувати, що дані залишаться узгодженими. Звичайно, з базою даних NoSQL транзакційність може бути в рамках одного документа або ви можете використовувати двофазі комміти, щоб отримати транзакційну семантику.
2. Втрата реляційної цілісності (зовнішні ключі): якщо в ваших дані є відносини, то вам доведеться застосовувати їх в додатку.
3. Відсутність можливості застосовувати структуру даних: строгі схеми іноді стають великою проблемою, але це також і потужний механізм надійного структурування даних, якщо грамотно їх використовувати. Документні БД, такі як MongoDB, забезпечують неймовірну гнучкість схеми, але ця гнучкість знімає відповідальність за збереження даних в чистоті.
4. Власну мову запитів: поява SQL стало абсолютною революцією, і з тих пір нічого не змінилося. Це неймовірно потужна мова, але і досить складна. Необхідність конструювати запити до БД на новій мові, що складається з фрагментів JSON, розцінюється як великий крок назад людьми, які мають досвід роботи з SQL

3.6 REST API

REST розшифровується як Representational State Transfer. Цеq термін, спочатку введений Роєм Філдінгом, який також був одним з творців протоколу HTTP. Відмінною особливістю сервісів REST є те, що вони дозволяють найкращим чином використовувати протокол HTTP.

Ресурс - це ключова абстракція, на якій концентрується протокол HTTP. Ресурс це все, що ви хочете показати зовнішньому світові через вашу програму. Важливо відзначити, що з REST вам потрібно думати про програму з точки зору ресурсів. Визначте, які ресурси ви хочете відкрити для зовнішнього світу Використовуйте

дієслова, вже визначені протоколом HTTP, для виконання операцій з цими ресурсами.

Ось як зазвичай реалізується служба REST:

1. Формат обміну даними: тут немає ніяких обмежень. JSON - дуже популярний формат, хоча можна використовувати і інші, такі як XML
2. Транспорт: завжди HTTP. REST повністю побудований на основі HTTP.
3. Визначення сервісу: не існує стандарту для цього, а REST є гнучким. Це може бути недоліком в деяких ситуаціях, оскільки додаток який використовує REST API необхідно розуміти формати запитів і відповідей. Однак для цього широко використовуються такі мови визначення веб-додатків, як WADL (Web Application Definition Language) і Swagger.

3.7 HTTP

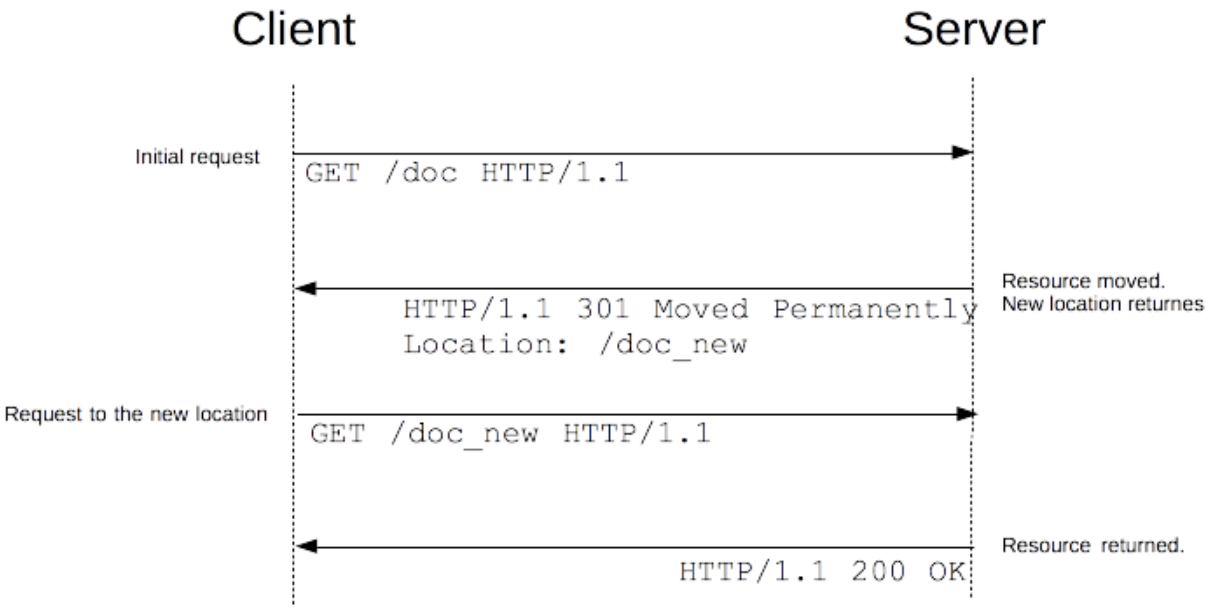
HTTP - це протокол який описує взаємодію між двома комп'ютерами (клієнтом і сервером), побудований на базі повідомлень, так званих запит (Request) і відповідь (Response). Кожне повідомлення складається з трьох частин: стартовий рядок, заголовки і тіло. При цьому обов'язковою є тільки стартовий рядок.

Тип HTTP-запиту (також званий HTTP-метод) вказує сервера на те, яку дію ми хочемо зробити з ресурсом. Спочатку передбачалося, що клієнт може хотіти від ресурсу тільки одне - отримати його, однак зараз по протоколу HTTP можна створювати пости, редагувати профіль, видаляти повідомлення і багато іншого. І ці дії складно об'єднати терміном “отримання”.

Для розмежування дій з ресурсами на рівні HTTP-методів і були придумані такі варіанти:

1. GET - отримання ресурсу.
2. POST - створення ресурсу.
3. PUT - оновлення ресурсу.

4. DELETE - видалення ресурсу.



3.8 Canvas

Canvas - це HTML елемент, який використовується для створення растрової графіки за допомогою JavaScript. Елемент <canvas> надає зручний API для малювання 2D графіки за допомогою JavaScript. На відміну від svg, canvas працює з растровою графікою. Це технологія миттєвого малювання, вона не зберігає свої елементи в дереві DOM, отже немає ніякого способу змінити існуючий малюнок або реагувати на події. Це означає, що, коли буде потрібно новий кадр, необхідно буде перемалювати всю сцену заново. Елемент <canvas> має тільки два атрибути - ширину і висоту. Якщо атрибути висоти і ширини не встановлені, то згідно специфікації html5 ширина елемента canvas буде дорівнює 300 пікселям, а висота 150. При зміні цих атрибутів canvas очищується. В системі canvas використовується для побудови графіків. Canvas відмінно підходить для створення інтерактивних зображень, що містять безліч складних деталей, градієнтних переходів і іншого подібного. Саме з

цієї причини canvas використовується при розробці в набагато більшої кількості ніж SVG

3.9 Mongoose

Mongoose - це ODM (Object Document Mapper). Це означає, що Mongoose дозволяє вам визначати об'єкти зі строго типізованою схемою, яка відповідає документу MongoDB. Mongoose надає величезний набір функціональних можливостей для створення і роботи зі схемами. На даний момент Mongoose містить вісім SchemaTypes (* типи даних схеми), які може мати властивість, яке зберігається в MongoDB. Ці типи наступні:

1. String.
2. Number.
3. Date.
4. Buffer.
5. Boolean.
6. Mixed.
7. ObjectId.
8. Array.

Для кожного типу даних можна:

1. Задати значення за замовчуванням .
2. Задати для користувача функцію перевірки даних вказати, що поле є обов`язковим.
3. задати get-функцію яка дозволяє вам проводити маніпуляції з даними до їх повернення у вигляді об’єкта.
4. задати set-функцію, яка дозволяє вам проводити маніпуляції з даними до їх збереження в базу даних визначити індекси для більш швидкого отримання даних.

3.10 SASS

Sass (Syntactically Awesome Stylesheets) — це скриптова метамова, яка компілюється в звичайні CSS-стилі.

У мови є два основних “діалекти»: SASS і новіший SCSS. Відмінності між ними невеликі, проте порушення правил синтаксису не дозволить скомпілювати файл. У SASS-синтаксисі немає фігурних дужок, вкладеність елементів в ньому реалізована за допомогою відступів, а стильові правила обов’язково відокремлені новими рядками.

Незалежно від синтаксису, SCSS назад сумісний з CSS. Тобто будь-який CSS обов’язково буде дійсним SCSS-кодом.

Sass дозволяє призначати змінні — і це одна з ключових переваг. Змінна, за аналогією з php, починається зі знака долара ($), значення присвоюються за допомогою двокрапки.

Змінні в Sass можна розділити на 4 типи:

1. число (int)
2. строка (string)
3. логічний тип (так/ні, boolean)
4. кольори (ім’я, імена)

4. ТЕЧНІЧНИЙ ОПИС СИСТЕМИ

4.1 Опис архітектури

Система включає в себе такі компоненти:

1. Веб-інтерфейс користувача у браузері.
2. Веб-сервіс для розрахунку моделі.
3. База даних для збереження результатів.
4. Сервер для веб систем.

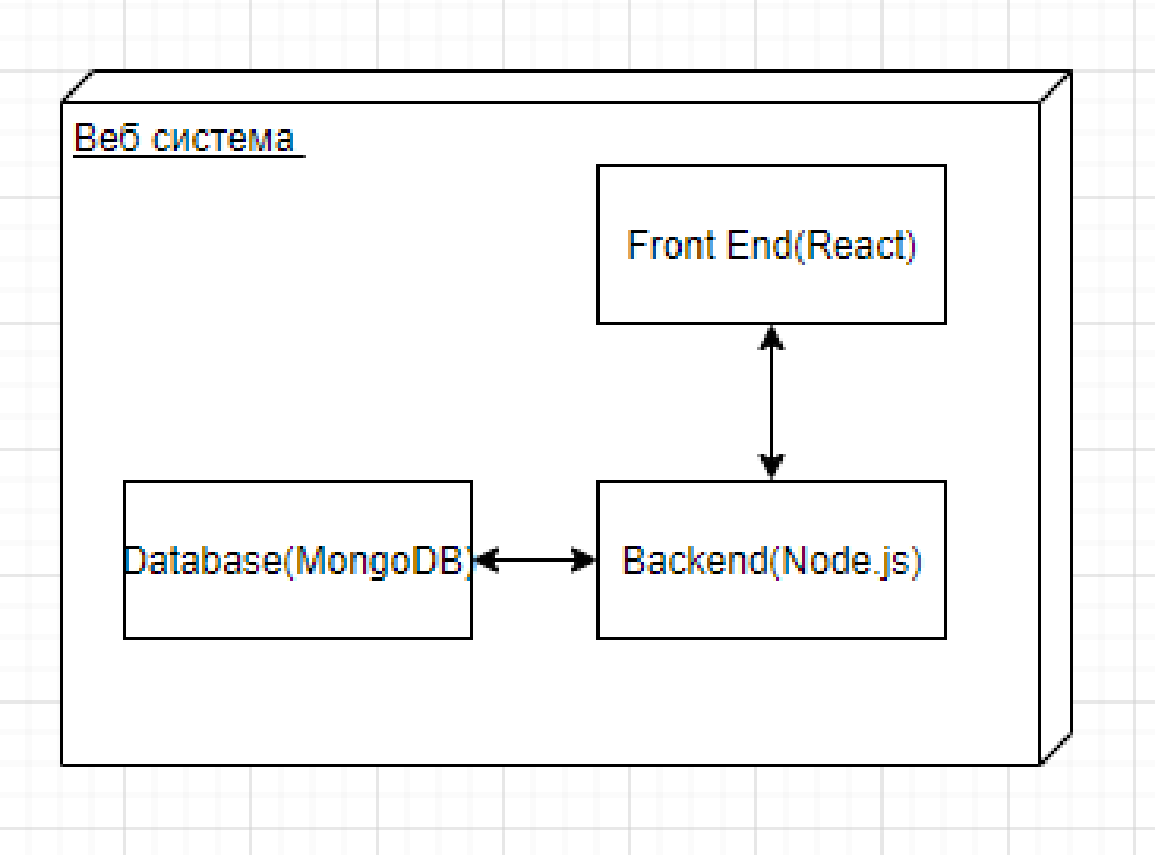


Рисунок 4.1 — Блок схема веб системи

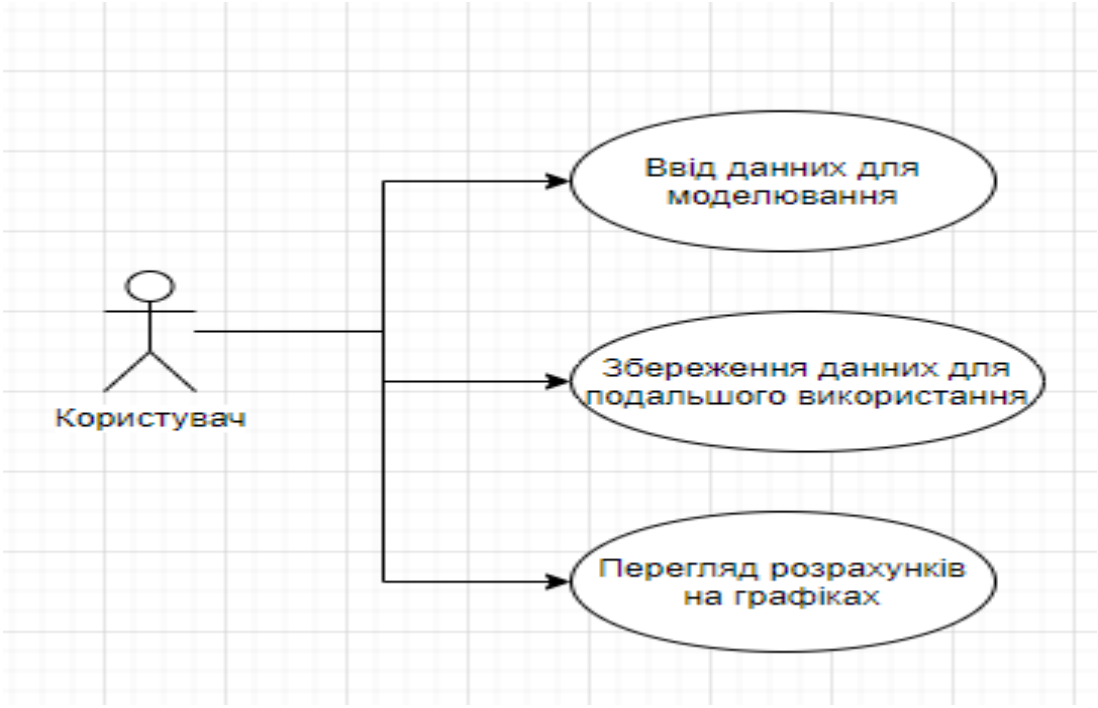


Рисунок 4.2 — Діаграма прецедентів системи

За Front End частину відповідальна бібліотека React, яка написана на мові програмування JavaScript.

За Backend частину відповідає Node.js, який можна легко маштабувати при більш великих навантажень на сервер

Щоб користувач скористався системою, він має мати доступ до Інтернету. Коли він входить на сайт, то в нього на екрані з’являється форма для внесення даних в яку він вводить вхідні данні такі, як:

1. Координати та глибина гідрофона (ГАС);
2. Характеристики об’єкту шуму (рухомий морський об’єкт)
   - 2.1 Стартове та кінцеве положення;
   - 2.2 Глибина;
   - 2.3 Швидкість;
   - 2.4 Амплітуда коливань;
   - 2.5 Частота хвил;
3. Геометричні розміри хвилі;

4. Час моделювання;
5. Частота коливання хвиль кожного об’єкта;
6. Амплітуда коливань хвилі;

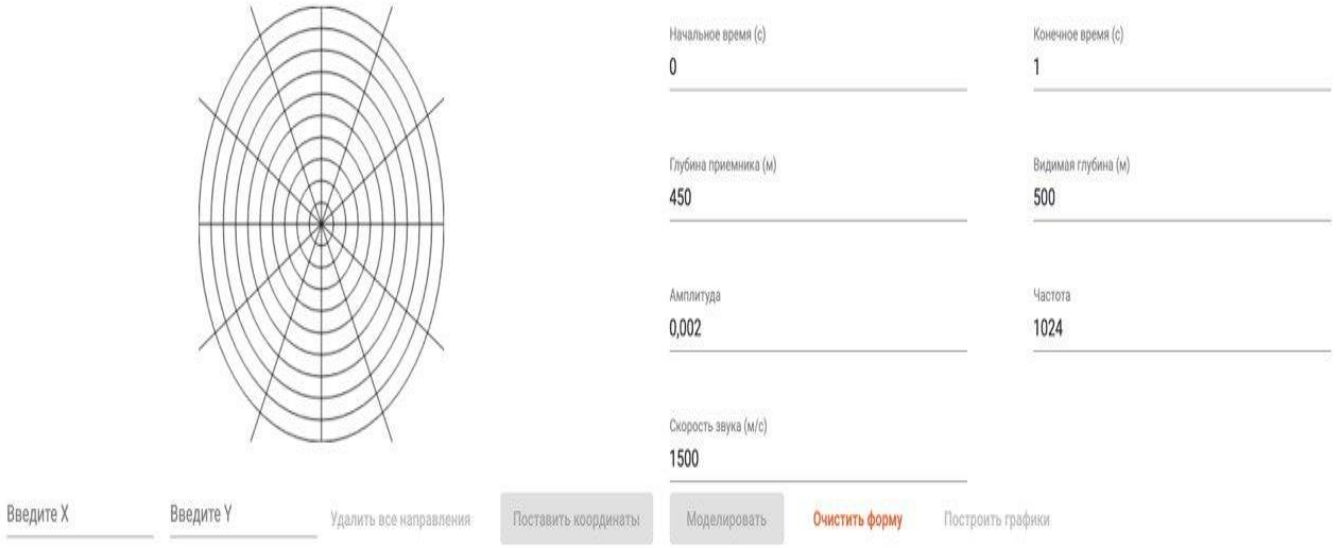


Рисунок 4.3 — Інтерфейс користувача

Також в нього є можливість зберегти ці данні для подальшого використання, натиснувши кнопку “Моделировать”, дані відправляються на сервер і зберігаються в базі даних. Коли сервер закінчить розрахунок, він відправить дані на веб-сторінку і користувач системи побачить Модальне вікно для скачування файлу з розрахунками в форматі json.

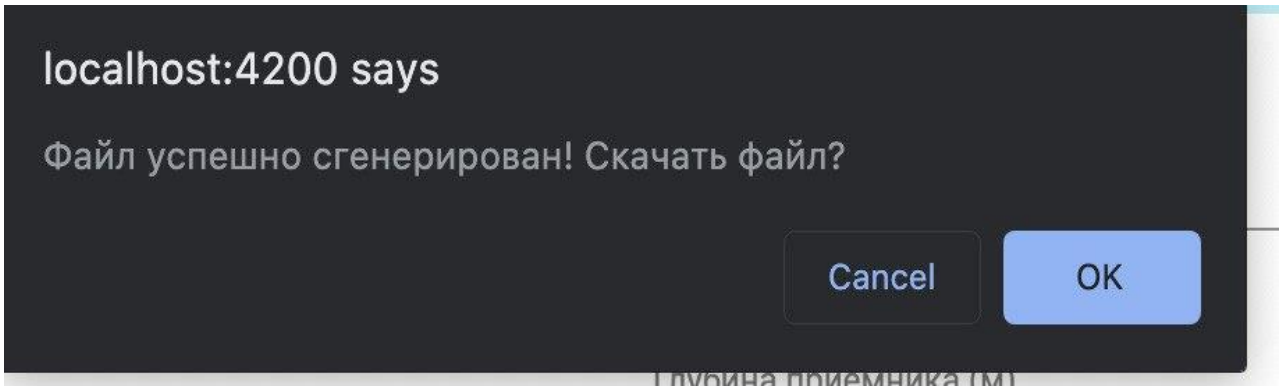


Рисунок 4.4 — Модальне вікно для завантаження файлу

Натиснувши на кнопку «OK» користувачу на комп’ютер завантажиться файл у форматі json, який вын зможе використати у подальшому для своіх цілей. Якщо користувач натисне на кнопку «Cancel», то модальне вікно зникне і файл не завантажиться, користувач зможе продовжити далі свою роботу з програмою

Після розрахунків користувач може натиснути на кнопку “Построить графики” і зможе отримати розрахунки у вигляді графіків

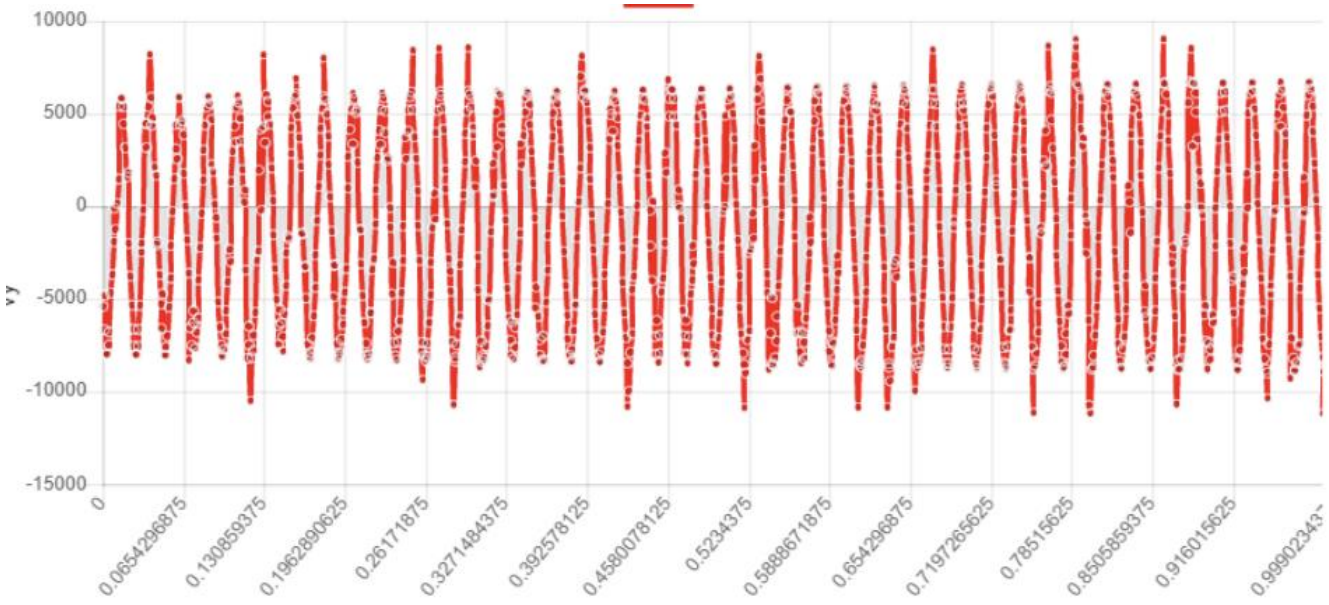


Рисунок 4.5 — Приклад графіка з відношенням векторної швидкості по X до часу

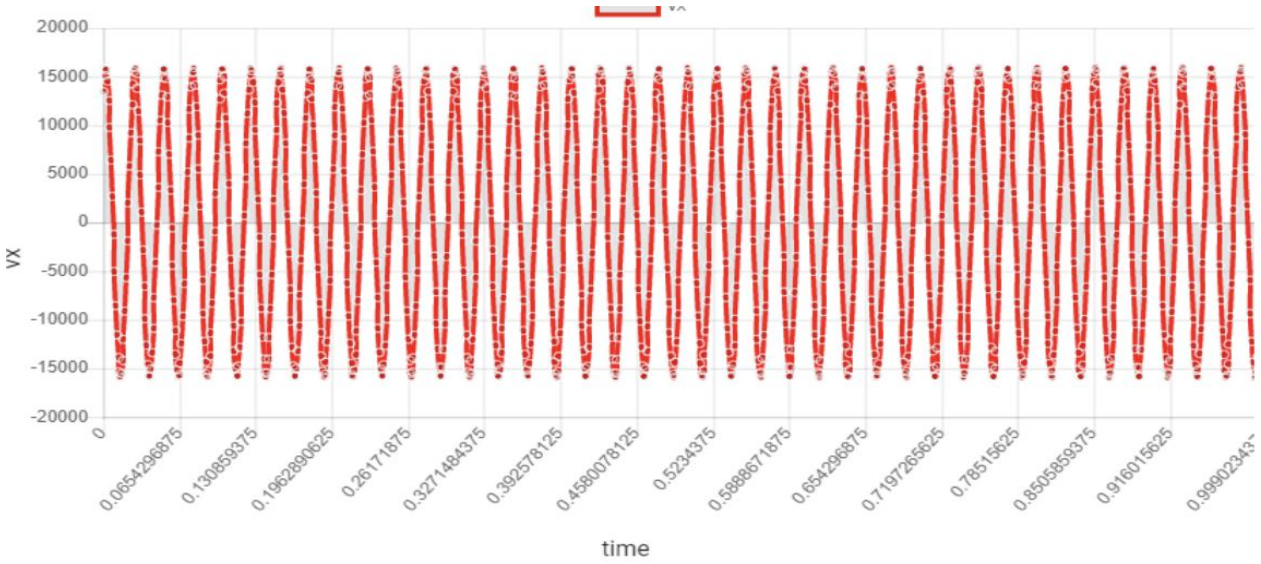


Рисунок 4.6 — Приклад графіка з відношенням векторної швидкості по Y до часу

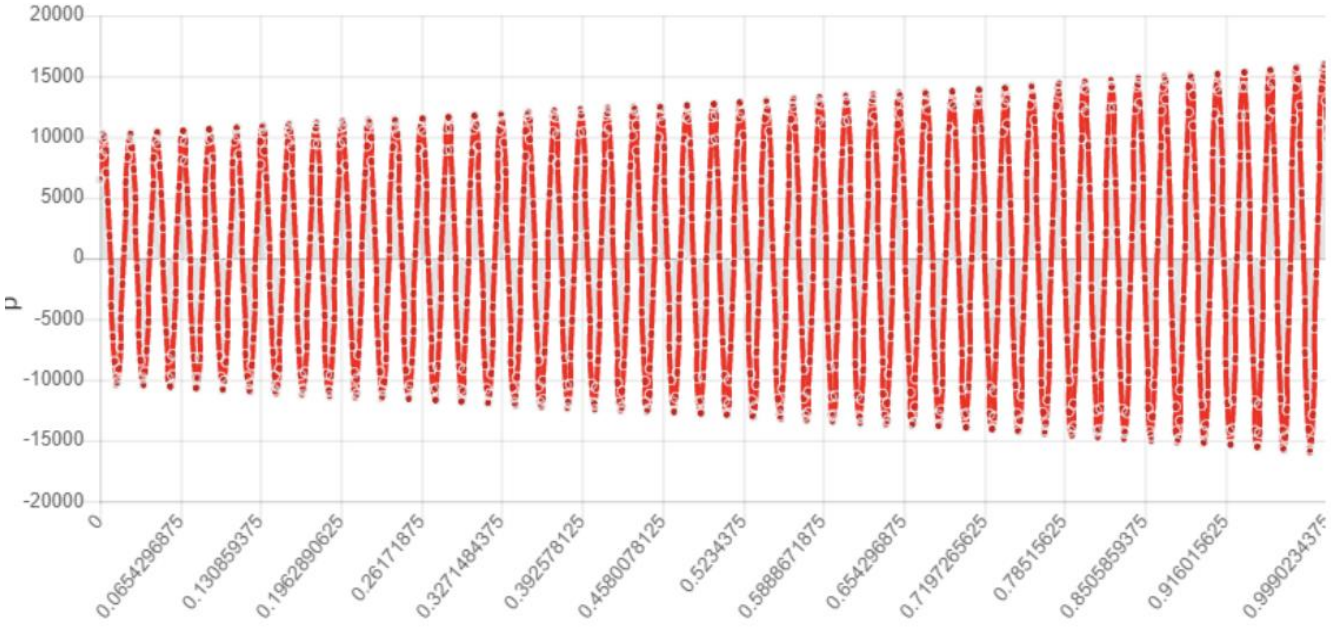


Рисунок 4.7 — Приклад графіка з відношенням векторної швидкості по Z до часу

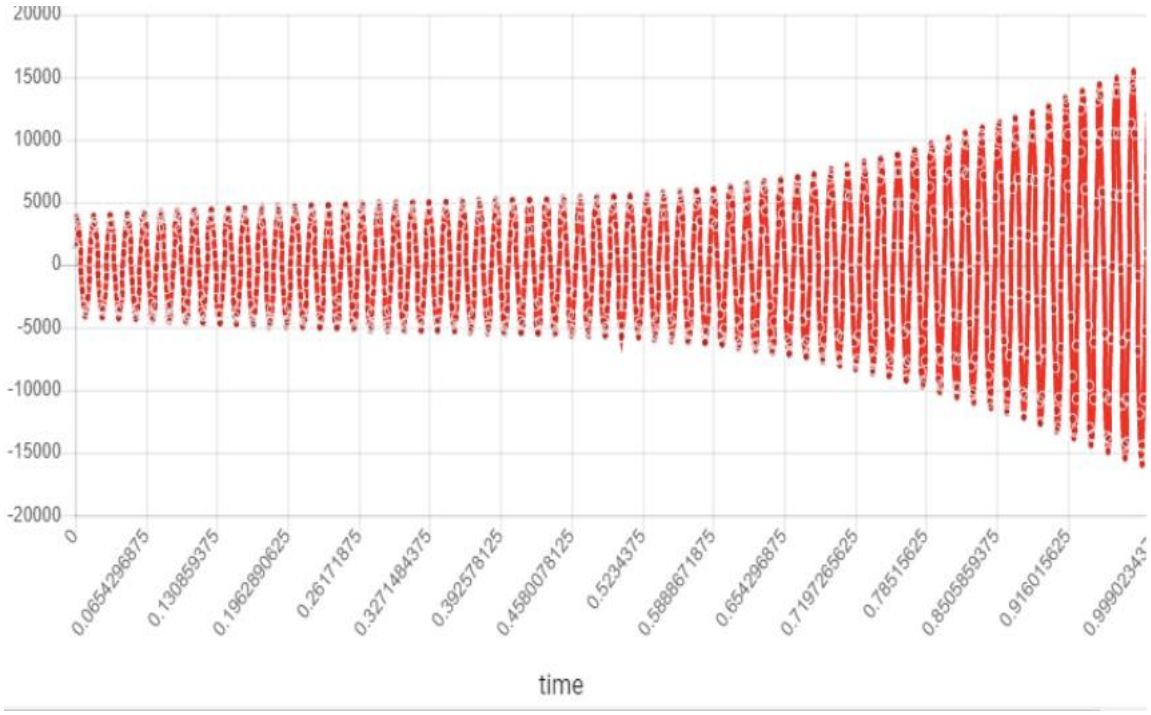


Рисунок 4.8 — Приклад графіка з відношенням тиску до часу

5. РОБОТА КОРИСТУВАЧА З СИСТЕМОЮ

Інтерфейс програми виконано за допомогою JSX яка є заміною HTML розмітки в React, CSS стилів та мови програмування Javascript з використанням бібліотеки React. Він включає в себе багато елементів керування для введення даних, вкладок і графіків для візуалізації інформації про водне середовище.

У правій частині користувацького інтерфейсу з формою для вводу початкових даних.

Вхідні дані:

1. Початковий час.
2. Кінцевий час.
3. Глибина гідрофона .
4. Глибина дна.
5. Амплітуда коливань променя.
6. Частота коливань.
7. Швидкість звуку.

Рисунок 5.1 — Форма для вводу вхідних даних

Потім користувачу потрібно вести координати об’єкту. Для цього є форма вводу координатів, так як напрям об’єкту задається через початкові і кінцеві координати, а також швидкість об’єкту

Рисунок 5.2 — Форма для вводу координатів об’єкта

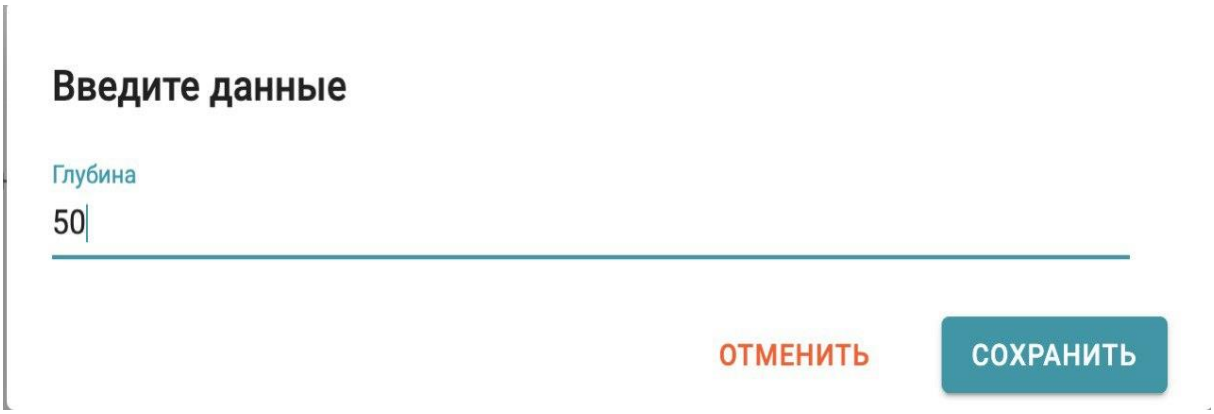


Рисунок 5.3 — Форма для вводу початкової глибини об’єкта

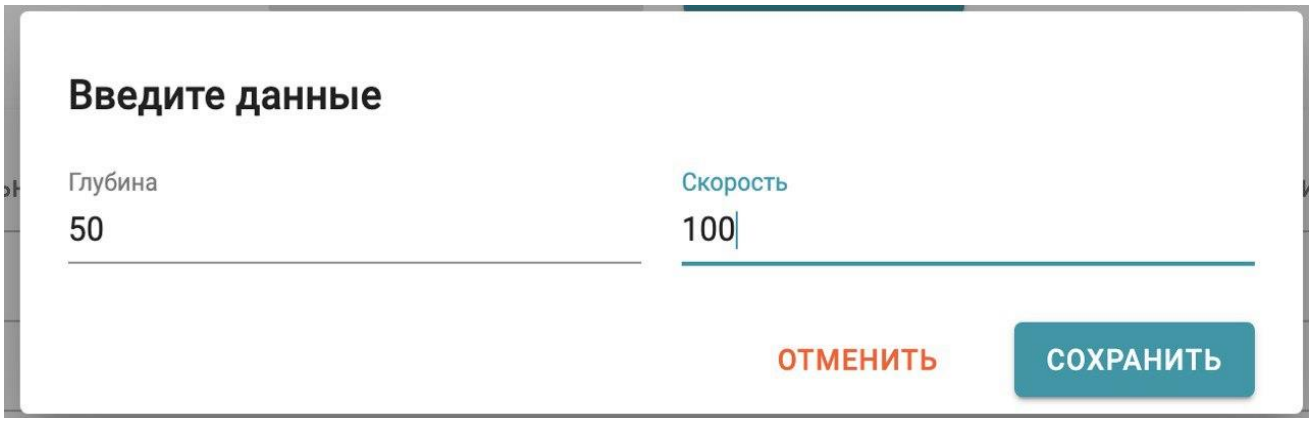


Рисунок 5.4 — Форма для вводу кінцевої глибини та швидкості об’єкта

При додаванні об’єкта характеристики додаються до списку об’єктів

Начальные координаты	Глубина начальной точки	Конечные координаты	Глубина конечной точки	Скорость
X: 11.000000000001364; Y: 11.000000000001364	50	X: 10000; Y: 10000	50	1
X: -13000; Y: 8000	50	X: 13000; Y: -8000	50	10
X: 0; Y: 12000	50	X: 0; Y: 12000	50	1
X: 0; Y: 12000	50	X: 0; Y: -12000	50	1

Рисунок 5.5 — Приклад списку з характеристиками об’єктів

При цьому ми можемо побачити направлення наших об’єктів на мапі

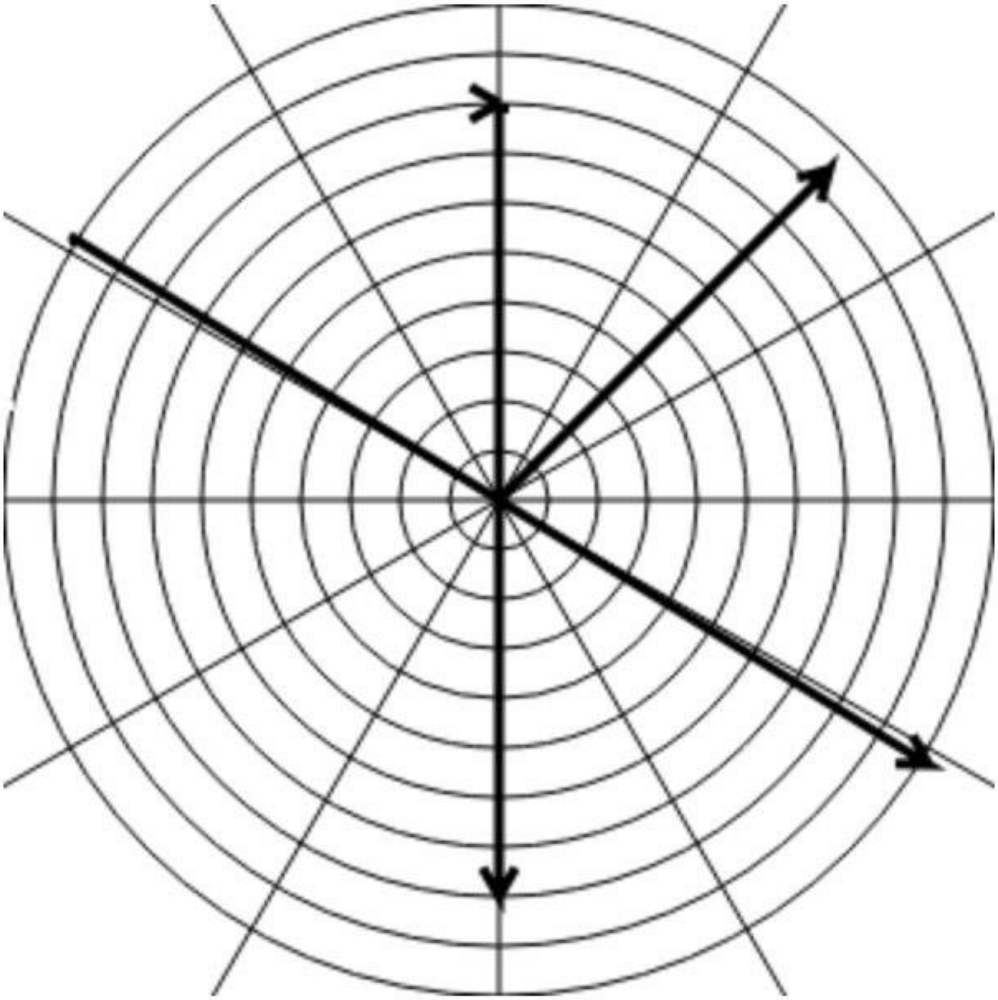


Рисунок 5.6 — Приклад направлення об’єктів на мапі

Також на мапі ми можемо побачити координати об’єктів при наведенні курсору на них

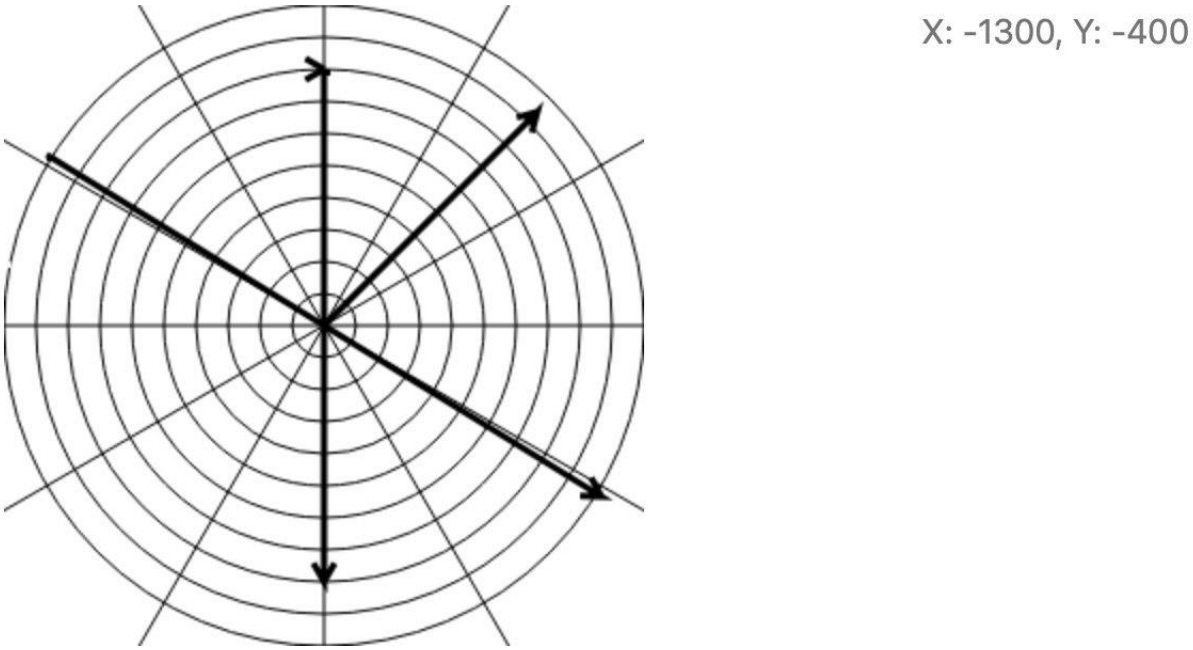


Рисунок 5.7 — Приклад зображення координатів об’єктів на мапі

Потім користувач натискає на кнопку “Моделировать” і посилає запит на сервер і сервер повертає файл для завантаження з результатами розрахунків

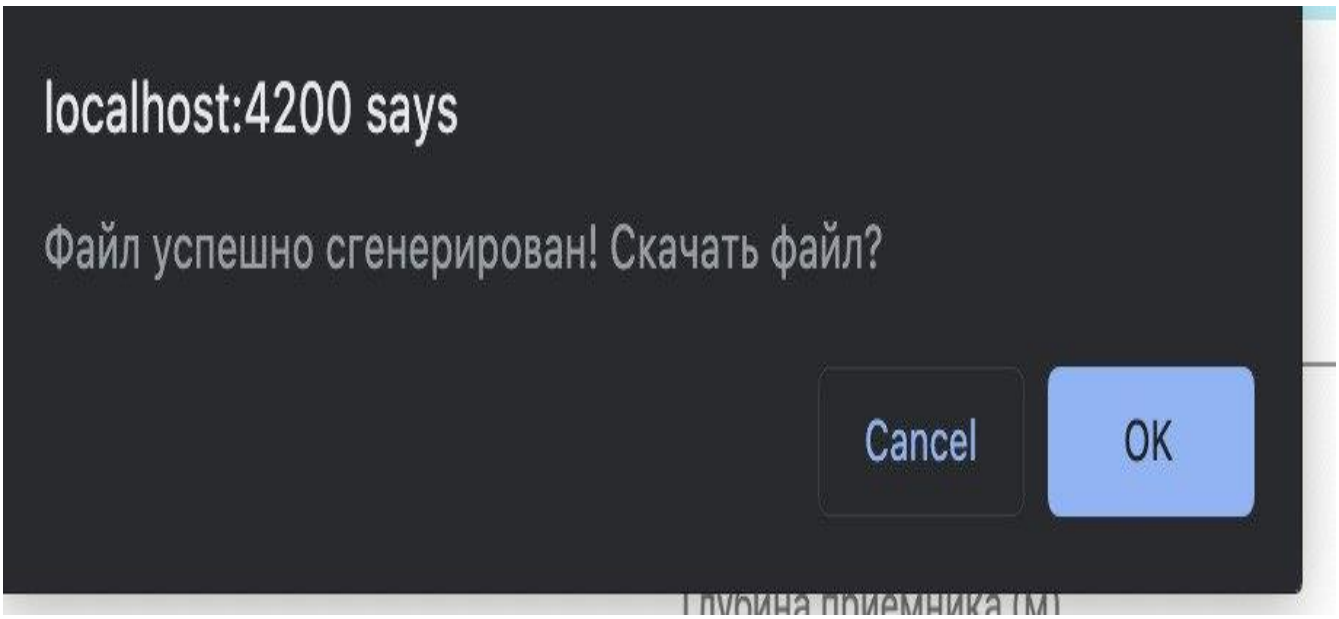


Рисунок 5.8 — Модальне вікно для завантаження файлу

Також користувач має можливість побачити результати розрахунків на графіках натиснувши кнопку “Построить графики”

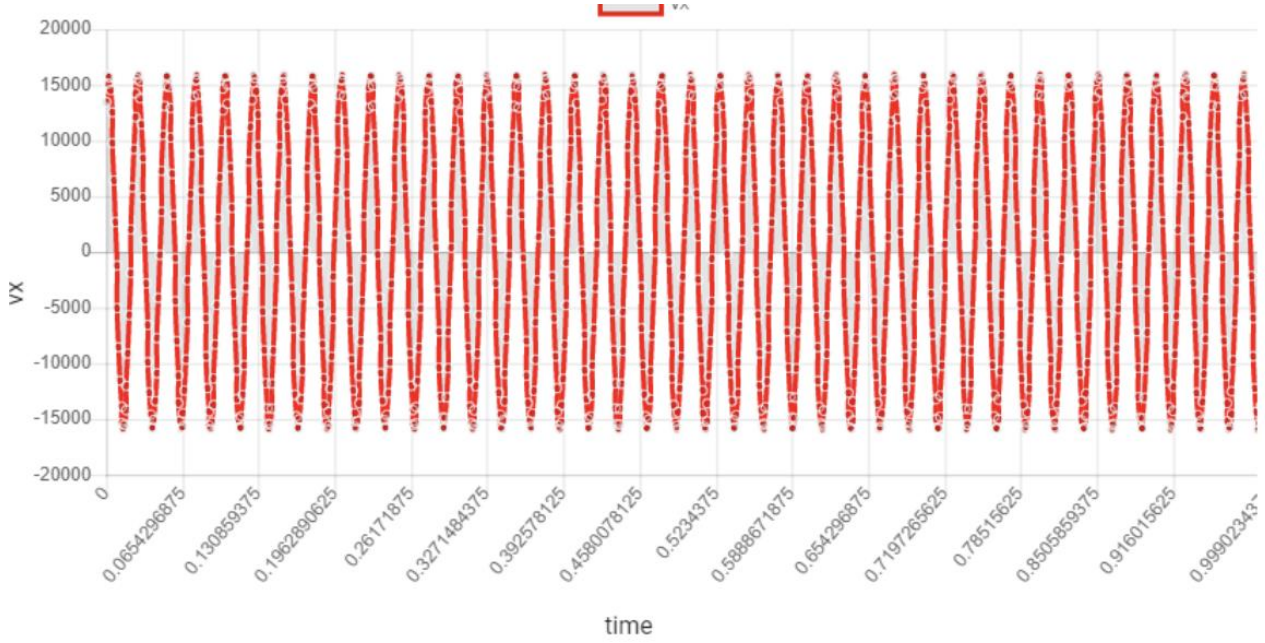


Рисунок 5.9 — Приклад графіка з відношенням векторної швидкості по X до часу

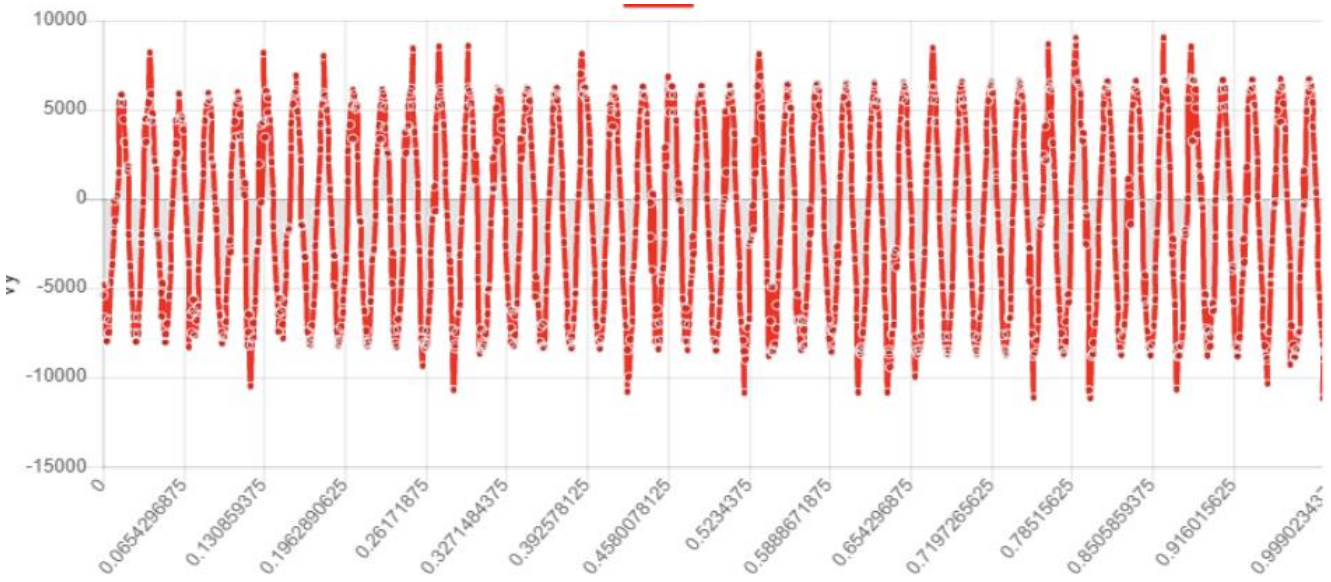


Рисунок 5.10 — Приклад графіка з відношенням векторної швидкості по Y до часу

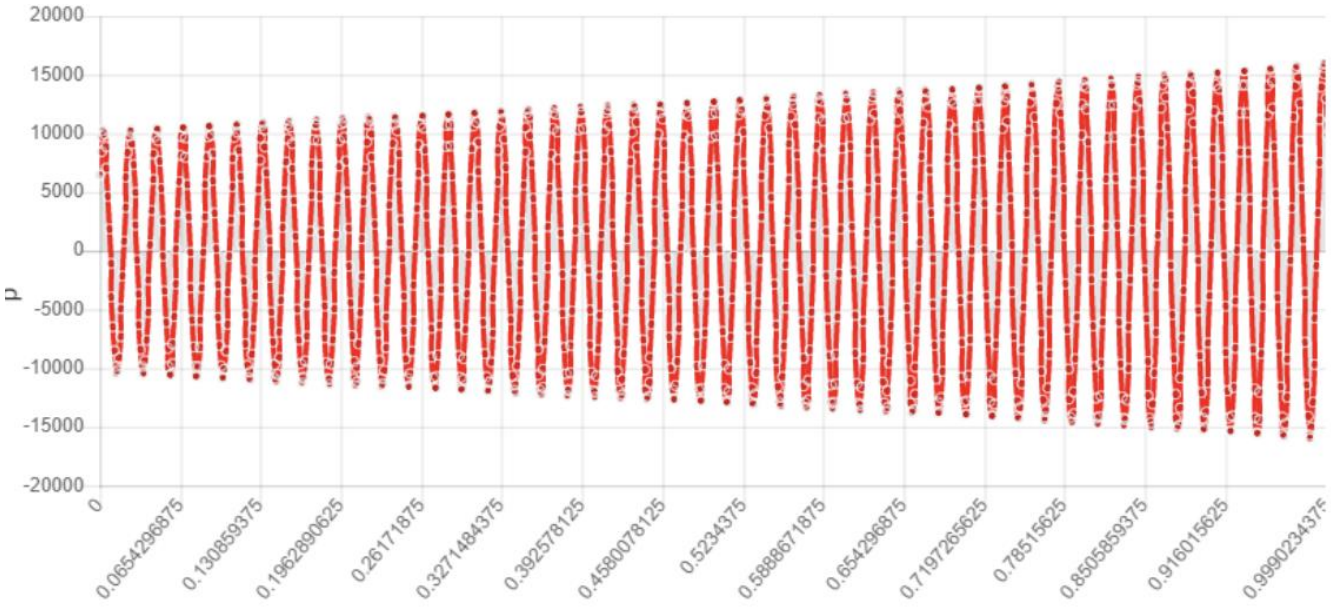


Рисунок 5.11 — Приклад графіка з відношенням векторної швидкості по Z до часу

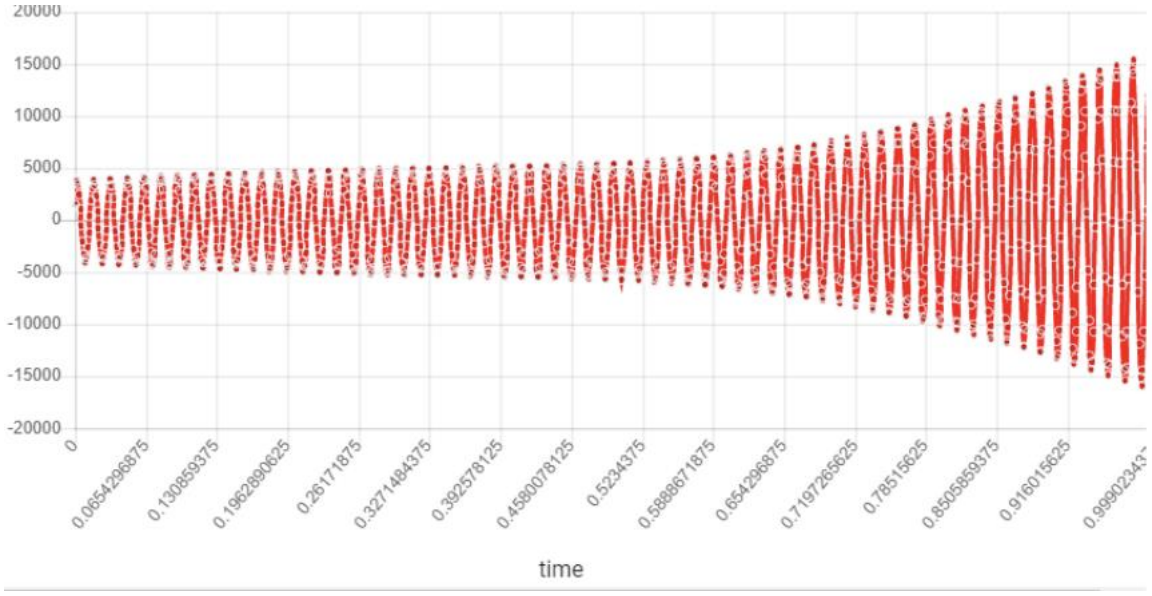


Рисунок 5.12 — Приклад графіка з відношенням тиску до часу

ВИСНОВКИ

За період практики, в ході виконання даної роботи було проведено ретельний аналіз предметної області, включаючи аналіз вимог до системи, вибір технологій, архітектуру веб-системи, організацію програмного коду. Результат – було створено веб-систему для моделювання гідроакустичного сигналу. Також в системі закладено фундамент для розробки нового функціоналу: інтеграція із сервісами ГІС; інтеграція з картами морів, рельєфів морів та океанів; додавання бізнес-логіки, яка б дозволяла організовувати розділення користувачів на користувача та адміністратора, щоб виконувати конфігурування системи; можливість зробити мобільний додаток для зручності в використані.

Розроблена веб-система стане початком для створення повноцінної системи для моделювання гідроакустичної ситуації з використанням різних джерел випромінювання звукових хвиль та гідрофонів. Вдосконалена система матиме можливість генерувати гідроакустичний сигнал і буде відображати на карті виявлені гідрофоном об’єкти. Можна буде обирати карти рельєфів морів та океанів, а також виставляти об’єкти на самій карті.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gulin O.E. and Yaroshchuk I.O. Simulation of underwater acoustical field fluctuations in range-dependent random environment of shallow sea // J. Comp. Acoust. 2014. Vol. 2
2. GIS Hydro’99 / Introduction to GIS Hydrology. — ESRI International User Conference (CD-R), 1999.
3. Francois, R. E., and G. R. Garrison. 1982. Sound absorption based on ocean measurements, 72(6), 1879–1890.
4. Распространение волн и подводная акустика / Под ред. Дж.Б. Келлера и Дж. Пападакиса. – М.: Мир, 1980. –230 с.
5. No.1, 1440006. 2. Бреховских Л.М., Лысанов Ю.П. Теоретические основы акустики океана. Наука, 2007. - 370 с.
6. В.С. Ковальский, А.Х. Дегтерев. Лучевая имитационная модель распространения звука в морской среде. НАН Украины. МГИ: – Севастополь. 2004. – 320 с.
7. Итан Браун Веб-разработка с применением Node и Express. - М. СПб: Издательский дом «Питер», 2016. - 336 с.
8. Документація Lucy Software [Електронний ресурс] – Режим доступу до ресурсу: http://oceansonics.com/lucy-software/
9. Eric Elliott Programming JavaScript Applications. - 1 edition изд. O'Reilly Media, 2014. - 254 с.
10. Документація HtmlBook [Електронний ресурс] – Режим доступу до ресурсу: http://htmlbook.ru/
11. Руководство Canvas [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/API/Canvas_API/Tutorial
12. Николас Закас ECMASCRIPT 6 ДЛЯ РАЗРАБОТЧИКОВ. Питер, 2017. - 352 с.
13. Basarat Ali Syed Beginning Node.js. Apress, 2014. - 308 с.
14. Документація по Node.Js [Електронний ресурс] – Режим доступу до ресурсу: https://Node.js.org/uk/docs/

Додаток 1

Веб система генерації гідроакустичного сигналу за променевою моделлю

Специфікація

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ62207_20Б

Аркушів 1

Київ – 2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ6 2207_20Б 81-1	Повєткін_Д_О_ТМ62.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ6 2207_20Б 12-1	Server.js	Модулі серверної частини
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ6 2207_20Б 12-2	app.component.js	Модуль клієнтської частини
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ6 2207_20Б 12-3	Modelling.js	Модуль моделювання сигналу
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ6 2207_20Б 12-4	PVCalculator.js	Модуль розрахунку векторної швидкості та тиску
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ6 2207_20Б 12-5	Rays.js	Модуль отримання променів
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ6 2207_20Б 12-6	FileSystem.js	Модуль для завантаження сигналу
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ6 2207_20Б 13-1	Опис.docx	Опис модуля інтерфейсу клієнтської частини програми

Додаток 2

Веб система генерації гідроакустичного сигналу за променевою моделлю

Лістинг програми

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ62207_20Б

Аркушів 13

Київ – 2020

```
// Код на запит з клієнтської сторони на моделювання
app.post('/api/modeling', (req, res) => {
   res.send({result: modelling.runModelling(req.body.params)});
});

// Код на запит з клієнтської сторони на завантаження
app.get('/api/download', (req, res) => {
   res.download('signal.json);
});
```



```
function runModelling(payload) {
   // Трансформація направлень в необхідний формат
   payload.arrows = transformObjects(payload.arrows);
   // Визначення кроку
   const timeStep = 1 / payload.frequency;

   // Ініціалізація модулю PVCalculator початковими даними
   pvCalculator.init({ ...payload, reflections: 5 });

   const list = [];
   for (let currentTime = payload.timeStart; currentTime < payload.timeEnd; currentTime
+= timeStep) {
      // Отримання даних про швидкість та тиск в кожен момент часу
      const pv = pvCalculator.getPV(currentTime);
      // Додавання їх у фінальний список в необхідному форматі
      list.push(getTimeSeriesGeneratorElement(currentTime, pv.pressure, pv.velocity));
   }
   // Збереження списку
   return save(list);
}

function save(list) {
```

```
    const MAX_VAL = 16000;
    const MIN_VAL = -16000;
    let minV = Number.MAX_VALUE;
    let maxV = -Number.MAX_VALUE;
    let minP = Number.MAX_VALUE;
    let maxP = -Number.MAX_VALUE;
    // Scale
    list.forEach(el => {
        if (el.pressure > maxP)
            maxP = el.pressure;
        if (el.pressure < minP)
            minP = el.pressure;
        maxV = (el.vx > maxV) ? el.vx : maxV;
        maxV = (el.vy > maxV) ? el.vy : maxV;
        maxV = (el.vz > maxV) ? el.vz : maxV;
        minV = (el.vx < minV) ? el.vx : minV;
        minV = (el.vy < minV) ? el.vy : minV;
        minV = (el.vz < minV) ? el.vz : minV;
    });
    const res = [];

    list.forEach(el => {
        // Отримуємо тиск
        const p = Math.round((el.pressure - minP) / (maxP - minP) * (MAX_VAL -
MIN_VAL) + MIN_VAL);
        // Отримуємо векторну швидкість по осі X
        const vx = Math.round((el.vx - minV) / (maxV - minV) * (MAX_VAL - MIN_VAL)
+ MIN_VAL);
        // Отримуємо векторну швидкість по осі Y
        const vy = Math.round((el.vy - minV) / (maxV - minV) * (MAX_VAL - MIN_VAL)
+ MIN_VAL);
        // Отримуємо векторну швидкість по осі Z
        const vz = Math.round((el.vz - minV) / (maxV - minV) * (MAX_VAL - MIN_VAL)
+ MIN_VAL);
        res.push({ p, vx, vy, vz, time: el.currentTime });
    });
    // Зберігаємо у файл через модуль файлової системи
    return fsModule.save(res);
```

```
}

function getTimeSeriesGeneratorElement(currentTime, pressure, velocity) {
   return {
      currentTime,
      pressure: pressure.z,
      vx: velocity.x,
      vy: velocity.y,
      vz: velocity.z
   }
}

function transformObjects(arrows) {
   const frequencies = [{ frequency: 40, amplitude: 1e-5 }];
   return arrows.map(arrow => ({ ...arrow, frequencies }))
}
```



```
function getPV(currentTime) {
   let res = {};

   objects.forEach(obj => {
      // Знаходження всіх променів відбиття в даний час
      const rays = rayTracer.getRays(getPosition(obj, currentTime), gasPosition,
seaDepth);
      obj.frequencies.forEach(freq => {
         rays.forEach(ray =>{
            // Дистанція від променя до вектора
            const r = getDistanceVector(ray);
            const dist = vectorModule.getLength(r);
            // Амплітуда
            const a = getAmplitude(freq.amplitude, dist);
            // Довжина хвилі
            const waveLen = getWaveLength(freq.frequency);
            // Вектор хвилі
            const k = getWaveVector(waveLen, r);
            // Зміна фази
```

```
            const phaseChange = getPhaseChange(freq.frequency, dist);
            // Швидкість
            const v = getVelocity(freq.frequency, a, k, r, currentTime, phaseChange);
            // Тиск
            const p = getPressure(freq.frequency, a, k, r, currentTime, phaseChange);
            res.velocity = res.velocity ? vectorModule.add(res.velocity, v) : v;
            res.pressure = res.pressure ? vectorModule.add(res.pressure, p) : p;
          });
      });
   });
   return res;
}

function getPosition(obj, time) {
   const beginLoc = vectorModule.getVector(obj.beginLocation.layerX,
obj.beginLocation.layerY, obj.beginLocation.depth);
   const targetLoc= vectorModule.getVector(obj.targetLocation.layerX,
obj.targetLocation.layerY, obj.targetLocation.depth);
   return vectorModule.add(beginLoc,
vectorModule.multiple(vectorModule.multiple(vectorModule.unitary(vectorModule.subtra
ct(targetLoc, beginLoc)), +obj.velocity), +time));
}

function getAmplitude(a0, dist) {
   return dist < Number.EPSILON
      ? a0
      : a0 * Math.exp(-amplitude * dist);
}

function getDistanceVector(ray) {
   const rayLen = ray.length;
   const n = ray.points.length;
   const dir = vectorModule.unitary(vectorModule.subtract(ray.points[n - 1], ray.points[n -
2]));
   return vectorModule.multiple(dir, rayLen);
}

function getWaveLength(waveFrequency) {
```

```
    return soundSpeed * waveFrequency;
}

function getWaveVector(waveLen, r) {
    const k = getWaveNumber(waveLen);
    return vectorModule.multiple(vectorModule.unitary(r), k);
}

function getWaveNumber(waveLen) {
    return 2 * Math.PI / waveLen;
}

function getPhaseChange(waveFrequency, distance) {
    return 2 * Math.PI * distance * waveFrequency / soundSpeed;
}

function getVelocity(frequency, a, k, r, t, phaseShift) {
    const w = frequency * 2 * Math.PI;
    const amp = w * a;
    const phi = vectorModule.multiple(k, r) - w * t + phaseShift + Math.PI;
    return getRotatedVectorWithLength(amp, phi, r);
}

function getPressure(frequency, a, k, r, t, phaseShift) {
    const w = frequency * 2 * Math.PI;
    const amp = r0 * soundSpeed * soundSpeed * vectorModule.getLength(k) * a;
    const phi = vectorModule.multiple(k, r) - w * t + phaseShift + Math.PI / 2;
    return getRotatedVectorWithLength(amp, phi, r);
}

function getRotatedVectorWithLength(max, phase, R)
{
    const xy = vectorModule.getVector(R.x, R.y, 0);
    const planeVector = vectorModule.getVector(vectorModule.getLength(xy), R.z, 0);
    const rotPlaneVector = vectorModule.rotateByZ(planeVector, phase);
    const resVector = vectorModule.multiple(vectorModule.unitary(xy), rotPlaneVector.x);
    resVector.z = rotPlaneVector.y;
    return vectorModule.multiple(vectorModule.unitary(resVector), max);
```

```
}
```



```
function getRays(objPos, gasPos, seaDepth) {
    if (!objPos)
        throw Error("objectPos must be not null");
    if (!gasPos)
        throw Error("gasPos must be not null");
    const res = [];
    const dirToObj = vectorModule.subtract(objPos, gasPos);
    const dirToObjXY = vectorModule.getVector(dirToObj.x, dirToObj.y);
    const relObj = vectorModule.getVector(vectorModule.getLength(dirToObjXY),
objPos.z);
    const relGas = vectorModule.getVector(0, gasPos.z);

    for (let reflections = -gReflections; reflections <= gReflections; reflections++)
    {
        // Find Relative Points
        const ray = getRelativeRayPoints(seaDepth, relObj, relGas, reflections);
        if (ray == null)
            continue;

        // Add other
        const toTopRealPoints = [];
        ray.forEach(relPoint => {
            const objWithoutDepth = vectorModule.add(gasPos,
vectorModule.multiple(vectorModule.unitary(dirToObjXY), relPoint.x));
            toTopRealPoints.push(vectorModule.getVector(objWithoutDepth.x,
objWithoutDepth.y, relPoint.y));
        });
        const toTopReal = trajectoryModule.getTrajectory(toTopRealPoints);



        res.push(toTopReal);
    }
    return res;
}
```

```
function getRelativeRayPoints(bottomDepth, start, end, reflections = 0) {
    if (!reflections) {
        return [ { ...start }, { ...end } ];
    }
    const isRayToTop = reflections <  0;
    reflections = Math.abs(reflections);
    if ((Math.abs(start.y) < Number.EPSILON && isRayToTop)
        || (Math.abs(bottomDepth - start.y) < Number.EPSILON && !isRayToTop)
        || (Math.abs(end.x - start.x) < Number.EPSILON && !reflections))
    {
        return null;
    }

    const res = [ { ...start } ];

    let bottomIsCurrentReflector = ((reflections % 2 === 1) && !isRayToTop) ||
(!(reflections % 2 === 1) && isRayToTop) ;

    const imageTo = { ...end };
    const images = [];
    for (let i = 0; i < reflections; i++) {
        // make reflection
        imageTo.y = bottomIsCurrentReflector
            ? bottomDepth + bottomDepth - imageTo.y
            : -imageTo.y;

        images.push({ ...imageTo });
        bottomIsCurrentReflector = !bottomIsCurrentReflector;
    }

    bottomIsCurrentReflector = !bottomIsCurrentReflector;

    let currentFrom = { ...start };
    for (let i = images.length - 1; i >= 0; i--)
    {
        // Current image
        let currentImageTo = images[i];
```

```
        // Find reflection point
        if (bottomIsCurrentReflector)
        {
            currentImageTo.x = currentFrom.x - currentFrom.x * (bottomDepth -
currentFrom.y) / (currentImageTo.y - currentFrom.y);
            currentImageTo.y = bottomDepth;
        }
        else
        {
            currentImageTo.x = currentFrom.x - currentFrom.x * currentFrom.y /
(currentFrom.y - currentImageTo.y);
            currentImageTo.y = 0;
        }


        currentFrom = currentImageTo;
        bottomIsCurrentReflector = !bottomIsCurrentReflector;
        res.push({ ...currentImageTo });
    }
    res.push({ ...end });

    return res;
}
```



```
// Збереження сигналу в файл
function save(res) {
    const response = writeBinary(res);
    fs.writeFile('./signal.json, response, e => {
        console.log(e || 'File successfully loaded.');
    });
    return res;
}
// Бінарний запис в файл
function writeBinary(res) {
    res.forEach(el => {
        bw.writeInt8(el.p);
```

```
      bw.writeInt8(el.vx);
      bw.writeInt8(el.vy);
      bw.writeInt8(el.vz);
    });
    return bw.toBuffer();
}
```



```
modeling(signalFormData: SignalFormInterface): void {
  const req = {
    ...signalFormData,
    arrows: this.arrows,
    pointWithoutDirection: this.pointWithoutDirection,
  };
  this.inProgress$.next(true);
  this.apiService.modeling(req).subscribe(
      res => {
          console.log(res);
          this.inProgress$.next(false);
          const shouldDownload = confirm('Файл успешно сгенерирован! Скачать файл?');
          if (shouldDownload) {
              const link = document.getElementById('download-link');
              link.click();
          }
          this.dataToBuildCharts = res;
      },
      e => alert(e.message || e),
  );
}


modeling(signalFormData: SignalFormInterface): void {
  // Запис в змінну для запиту з правильною структурою
  const req = {
    ...signalFormData,
    arrows: this.arrows,
    pointWithoutDirection: this.pointWithoutDirection,
```

```
  };
  this.inProgress$.next(true);
  // Звернення до API сервісу для запиту на початок моделювання
  this.apiService.modeling(req).subscribe(
    // Успішний сценарій моделювання
    res => {
      this.inProgress$.next(false);
      const shouldDownload = confirm('Файл успешно сгенерирован! Скачать файл?');
      if (shouldDownload) {
        // Початок завантаження файлу з сервера
        const link = document.getElementById('download-link');
        link.click();
      }
      // Збереження даних про сигнал
      this.dataToBuildCharts = res;
    },
    // Обробка помилки
    e => alert(e.message || e),
  );
}

// Метод для посилання запиту на серверну частину для початку моделювання
modeling(payload: ModelingInterface): Observable<ModellingResponseDataInterface[]>
{
  return
this.http.post<ModellingResponseDataInterface[]>('http://localhost:8000/api/modeling', {
params: payload, observe: 'response' }).pipe(
    pluck('result'),
  );
}


// Створення форми для введення початкових даних з заданими дефолтними даними
private createForm(): void {
  this.signalForm = this.fb.group({
    timeStart: [
      0,
      Validators.required,
```

```
    ],
    timeEnd: [
      1,
      Validators.required,
    ],
    gasDepth: [
      450,
      Validators.required,

    ],
    seaDepth: [
      500,
      Validators.required,

    ],
    amplitude: [
      0.002,
      Validators.required,
    ],
    frequency: [
      1024,
      Validators.required,
    ],
    soundSpeed: [
      1500,
      Validators.required,
    ],
  });
}
```

```
< <!--Загльна структура системи-->

<!--Контент-->

<div class="global-content">

    <div class="d-flex">

        <div class="flex-1">
```

```
            <!--Компонент з головною формою-->
            <app-input-form></app-input-form>
        </div>
        <div class="flex-1">
            <!--Компонент з акваторією-->
            <app-input-chart></app-input-chart>
        </div>
    </div>
    <!--Компонент для виводу таблиці з направленнями-->
    <app-input-appearence></app-input-appearence>
</div>

<!--Невидима силка для скачування згенерованого-->
<a style="display: none" id="download-link" href="http://localhost:8000/api/download" download></a>
```

Додаток 3

Веб система генерації гідроакустичного сигналу за променевою моделлю

Опис програмного коду

УКР.НТУУ«КПІ ім. Ігоря Сікорського» ТЕФ_АПЕПС_ТМ62207_20Б 13-1

Аркушів 7

Київ – 2020

Анотація

Додаток надає можливість моделювати та генерувати гідроакустичні сигнали за променевою моделлю.
Розроблене програмне забезпечення дозволяє отримати візуальне представлення сигналу у вигляді графіку та звукове представлення у вигляді файлу json розширення після введення початкових даних.
Веб система була розроблена за допомогою платформи VS Code з використанням мови Node.js для серверної сторони програми та мови Javascript з бібліотекою React для клієнтської.

ЗМІСТ

-3-

Загальні відомості

Відповідно до теми дипломної роботи, програма має назву Система генерації гідроакусничного сигналу».

Програма працює через будь-який браузер та потребує доступу до мережі інтернет, але не потребує встановлення на ПК зайвого ПО, що забезпечує зручність та швидкість.

Система була написана мовою Javascript з використанням бібліотеки React та Node.js.

-4-

Функціональне призначення

Розроблений програмний засіб повинен вирішити задачу виявлення рельєфу морського дна, навігації під водою тощо. Це було реалізовано за допомогою кількох функцій системи, а саме:

- моделювання гідроакустичного сигналу за допомогою розробленого алгоритму;
- завантаження згенерованого файлу з розширенням json;
- відображення графічної візуалізації гідроакустичного сигналу

-5-

Опис логічної структури

Програмний продукт складається з клієнтської та серверної частини.
Загальний принцип роботи додатку такий:

1. Користувач вводить початкові дані;
2. Натискає на кнопку “Моделировать”
3. Метод, що викликається в обробнику виконує запит;
4. Результати виконання методу виводяться на сторінку.

В першу чергу завантажується форма з усіма її компонентами: кнопки, текстові поля, таблиці та акваторія. Користувач вводить всі необхідні дані та додає направлення на акваторії.
Після натискання на кнопку “Моделировать”, що розміщена під формою, викликається метод modelling, в який передаються всі початкові параметри. Він виконує запит до сервера та передає всі ці дані. Сервер в свою чергу починає проведення розрахунків за написаним алгоритмом. Після успішного завершення він повертає результат на клієнтську сторону у вигляді масиву з порахованими показниками сигналу для його візуалізації та силку на скачування згенерованого файлу з json розширенням.
Після цього користувач може завантажити даний сигнал через діалогове вікно завантаження браузера, яке відкриється одразу після генерації, або подивитись візуалізацію сигнала, натиснувши на конпку “Посмотреть графики”.
Після закінчення роботи з цими даними, користувач без перезавантаження сторінки може очистити всі введені початкові дані і направлення відповідними кнопками та почати генерацію наступного сигналу.

-6-

Використовувані технічні засоби

Для організації доступу до програмного продукту потрібно мати комп’ютер або ноутбук.
Кінцевим користувачам для роботи з програмою потрібно, щоб на комп’ютері був встановлений будь-який браузер, Node.js та React scripts.

-7-

Вхідні і вихідні дані

Вхідними даними є:

- глибина водного середовища;
- початковий час;
- кінцевий час;
- глибина гідрофона;
- частота коливань;
- амплітуда коливань;
- направлення та відомості про них.

Вихідними даними є:

- згенерований гідроакустичний сигнал у json файл;
- візуалізація сигналу у вигляді графіка.

Додаток 4

Веб система генерації гідроакустичного сигналу за променевою моделлю

Довідки про впровадження результатів роботи

УКР.НТУУ«КПІ ім. Ігоря Сікорського» ТЕФ_АПЕПС_ТМ62207_20Б 13-2

Аркушів 1

Київ – 2020

“Затверджую”

В.о. зав. кафедри АПЕПС
КПІ ім. Ігоря Сікорського
к.т.н. Коваль О.В.

“ ” червня 2020 р.

АКТ ВПРОВАДЖЕННЯ

результатів дипломної роботи освітньо-кваліфікаційного рівня “бакалавр”
Повєткіна Дмитро Олексійовича

Дмитро ПОВЄТКІН в процесі виконання дипломної роботи на тему “ *Веб система генерації гідроакустичного сигналу за променевою моделлю* ” розробив WEB-систему, якиа дозволяє формувати уніфікований опис умов проведення гідроакустичного експерименту, визначати відповідну сцену експерименту та сценарій проведення експерименту, виконувати моделювання та виводити на WEB-сторінку результат моделювання, а також зберігати результати моделювання у файлі формату Json для подальшої обробки.

Крім того, зазначений програмний продукт є складовою програмного комплексу з моделювання гідроакустичних процесів, якій розробляється у процесі виконання ініціативної науково-дослідної роботі на тему «*Методи моделювання гідроакустичних сигналів морських об'єктів*» що виконується на кафедрі АПЕПС ТЕФ у межах робочого часу (номер державної реєстрації теми :0120U100973).

Даний програмний продукт приймається до дослідної експлуатації в науково навчальній лабораторії з моделювання динамічних процесів та систем

Науковий керівник теми № 0120U100973
к.т.н., доцент Іван ВАРАВА
Завідуючій лабораторією Володими ГАЙДАРЖИ